INTERWOVEN

TeamSite®
Command-Line Tools

Release 5.5.1

For Windows NT® and Windows® 2000

INTERWOVEN

Table of Contents

Chapter 3: Development Tools   71

iwcallback.exe 89
iwcat.exe 90
iwckrole.exe 91
iwcmp.exe 92
iwdecode.exe 94
iwdeploy.exe 95
iwencode.exe 96
iwevents.exe 97
iwextattr.exe 100
iwgetwfobj.exe 102
iwinvokejob.exe 109
iwjobc.exe 110
iwjobvariable.exe 111
iwlasted.exe 112
iwlist.exe 113
iwlistlocks.exe 115
iwlistmod.exe 116
iwlock.exe 117
iwlockinfo.exe 118
iwmerge.ipl 119
iwmkbr.exe 120
iwmkwa.exe 121
iwnexted.exe 122
iwprv.exe 123
iwpublish.exe 124
iwqueryjobs.exe 125
iwquerytasks.exe 126
iwrename.exe 128
iwretryjobop.exe 129
iwrevert.exe 130
iwrlog.exe 131
iwrmbr.exe 133
iwrmed.exe 134
iwrmjob.exe 135
iwrmtaskfile.exe 136

Chapter 4: Command Triggers   153

Appendix A: Master List   175

About This Book

Content and Audience

TeamSite Command-Line Tools describes each TeamSite® command-line tool (CLT) and command trigger, including syntax and usage examples. It is intended primarily for TeamSite Administrators and Master users, and for web server administrators and system administrators. Users of this manual should be familiar with either IIS or Netscape web servers, and with basic Windows NT operations such as adding users and modifying ACLs.

How This Manual is Organized

This manual is organized as follows:

- Chapter 1, "Overview of CLTs and Command Triggers" - Describes the types of operations you can perform with CLTs and command triggers, how CLTs and command triggers are grouped into functional categories, and the general syntax you must use when entering information (for example, vpaths and object IDs) common to many CLTs.

- Chapter 2, "Administration Tools" - Contains man pages for the System Information, System Services, and Backing Store CLTs. All man pages in the chapter are presented as one group, arranged alphabetically.

- Chapter 3, "Development Tools" - Contains man pages for CLTs in the following categories: General Development, Deployment, Branch Operation, Edition Operation, Workarea Operation, Version Management, Workflow/Job, and Templating. All man pages in the chapter are presented as one group, arranged alphabetically.

- Chapter 4, "Command Triggers" - Contains man pages for command triggers from the following groups: General Events, File and Directory Events, Branch Events, Edition Events, and Workarea Events. All man pages in the chapter are presented as one group, arranged alphabetically.
- Appendix A, "Master List"- Contains a single alphabetized list of all CLTs and command triggers described in this manual.
- Appendix B, "Sample Command Trigger Scripts" - Contains examples of scripts that can be triggered by command triggers.
- Appendix C, "Error Codes" - Lists the error codes that can be returned by CLTs.

Notation Conventions

This manual uses the following notation conventions:

Convention	Definition and Usage
Bold	Text that appears in a GUI element (for example, a menu item, button, or element of a dialog box) and command names are shown in bold. For example: Click **Edit File** in the Button Bar.
Italics	Book titles appear in italics. Terms are italicized the first time they are introduced. Valuable information may be italicized for emphasis.
`Monospace`	Commands, command-line output, and file names are in monospace type. For example: The `iwextattr` command-line tool allows you to set and look up extended attributes on a file.

Convention	Definition and Usage
Monospaced italic	Monospaced italics are used for command-line variables. The most common example of this is *iw-home*, which refers to the directory where TeamSite is installed. For example: *iw-home*\etc\iw.cfg is the path to the main TeamSite configuration file, iw.cfg, which is located in the etc directory under the TeamSite installation directory. iwckrole *role user* means that you must insert the values of *role* and *user* yourself.
Monospaced bold	Monospaced bold represents information you enter in response to system prompts. The character that appears before a line of user input represents the command prompt, and should not be typed. The **>** character that appears before a line of user input represents the command prompt, and should not be typed. For example: **>iwextattr -s project=proj1 //IWSERVER/default/ main/dev/WORKAREA/andre/products/index.html**
Monospaced bold italics	Used to indicate a variable in response to a system prompt.
[ ]	Square brackets surrounding a command-line argument mean that the argument is optional.
{ }	Curley braces surrounding a command-line argument mean that the argument is required.
|	Vertical bars separating command-line arguments mean that only one of the arguments can be used.

Chapter 1

Overview of CLTs and Command Triggers

CLTs enable you to use the command line instead of the browser or file system interface to perform most TeamSite administration and development tasks. For example, TeamSite provides CLTs for creating and deleting branches and workareas, publishing, deploying, assigning files, invoking the proxy server, interacting with jobs, and dozens of other tasks. CLTs are designed for users and administrators who are comfortable using a command-line interface, and who cannot or prefer not to use the browser or file system interface.

In addition to CLTs, TeamSite also supports a set of command triggers that let you configure TeamSite to execute custom scripts whenever certain events occur. For example, you can use command triggers to execute an email notification script whenever a file is assigned, or whenever a new branch is created, or when any of several other supported events occur. The scripts that are triggered can be simple or complex and typically are created by an administrator for a specific site or installation.

You should use CLTs and command triggers only if you have a good understanding of TeamSite structure, roles, and concepts. When learning CLTs, it is often helpful to monitor your activities using the GUI. If you choose to monitor from the GUI, be sure to refresh your view often. In some cases, it might be helpful to run multiple TeamSite sessions using the GUI.

The following sections briefly describe the default set of TeamSite CLTs and command triggers.

INTERWOVEN

CLTs

CLTs are organized into two main categories: administration tools and development tools.

Administration Tools

Administration tools are divided into the following categories:

- **System Information**: Returns system-wide information or information about a file's location. They do not provide information about the data in a file (the development tool CLTs provide that type of information).
- **System Services**: Enable you to manipulate files or objects that affect the entire TeamSite system. Typical tasks include, but are not limited to, editing configuration files, performing system backups, and starting the proxy server.
- **Backing Store**: Enables you to perform various tasks on the backing store, including checking for and fixing problems, converting to and from a multiple file system backing store, and merging metadata files.

See Chapter 2, "Administration Tools," for details about the CLTs in each category.

Development Tools

Development tools are divided into the following categories:

- **General Tools**: Enables you to perform operations on, or retrieve information about, the data in a specific file or TeamSite object.
- **Deployment**: Enables you to you deploy a TeamSite edition to a production server.
- **Branch Operations**: Enable you to you create and manipulate TeamSite branches.
- **Edition Operations**: Enable you to you create and manipulate TeamSite editions.
- **Workarea Operations**: Enable you to create and manipulate TeamSite workareas.
- **Version Management**: Returns information about file revisions and versions, and let you manipulate specific versions of files and directories.
- **Workflow/Job**: Enables you to control workflow and job elements such as file locks, assignments to authors, and file approval/rejection.
- **Templating**: Enables you to update, insert data into, and otherwise manipulate template-based files. Available only if TeamSite Templating is installed.

See Chapter 3, "Development Tools," for details about the CLTs in each category.

Command Triggers

All command triggers are described in Chapter 4, "Command Triggers." Two sample command trigger perl scripts—one that sends email notification and one that replicates files—are shown in Appendix B, "Sample Command Trigger Scripts."

Before Using CLTs

Most CLTs require that you specify some type of information when you invoke the command. For example, you might have to specify an object ID or the location of a TeamSite object, or provide some other type of information. The following sections describe how to provide this information.

Additionally, you can enter the name of the CLT followed by `-h` to obtain "help" on that CLT. The help provides a brief description of the CLT and lists the flags and options for that CLT. You can enter the CLT name followed by `-v` to display version information. If `-h` or `-v` are specified, any other flags or options entered at the same time are ignored.

Version Paths

Most CLTs require that you use a *version path* (vpath) to specify the location of a branch, workarea, staging area, edition, file, or directory in TeamSite. Vpaths are only used for items within TeamSite; all other files (for example, configuration files) must be specified with a directory path (see page 18). Vpaths have the following form:

To specify a server:

`//servername`

For example, the default vpath for the TeamSite server is:

`//IWSERVER`

To specify an archive:

//servername/archivename

For example, the default vpath for the TeamSite archive is:

    //IWSERVER/default

To specify a branch or sub-branch:

//servername/archivename{[/branchname]+}

//servername/archivename{[/branchname]+}/subbranchname

For example, the default vpath for the main branch is:

    //IWSERVER/default/main

The vpath for a sub-branch named `dev` off of the main branch would be:

    //IWSERVER/default/main/dev

To specify an edition:

//servername/archivename{[/branchname]+}/EDITION/*editionname*

For example, the default vpath for the initial edition on the main branch is:

    //IWSERVER/default/main/EDITION/INITIAL

The name of an edition on the subbranch `dev` would be:

    //IWSERVER/default/main/dev/EDITION/ed_0001

To specify a workarea:

`//servername/archivename{[/branchname]+}/WORKAREA/workareaname`

For example, the vpath for a workarea named `eng` on the main branch would be:

    //IWSERVER/default/main/WORKAREA/eng

The vpath for a workarea named `qa` on the sub-branch `dev` would be:

    //IWSERVER/default/main/dev/WORKAREA/qa

To specify an individual file within a workarea:

`//servername/archivename{[/branchname]+}/WORKAREA/workareaname/`
`directory-pathname/filename`

For example, the vpath for a file named `logo.gif` in the directory `/htdocs/gifs` in workarea `eng` on sub-branch `dev` would be:

    //IWSERVER/default/main/dev/WORKAREA/eng/htdocs/gifs/logo.gif

Relative vpaths

Most TeamSite CLTs only require you to specify relative vpaths. For example, instead of specifying `//IWSERVER/default/main` you would only need to specify `main`.

In the following example, instead of specifying:

    //IWSERVER/default/main/dev/WORKAREA/eng/htdocs/gifs/logo.gif

you would only need to specify:

    main/dev/WORKAREA/eng/htdocs/gifs/logo.gif

If you are in the `htdocs` directory of workarea `eng` on the `main` branch, you only need to specify:

    gifs/logo.gif

INTERWOVEN

Directory Paths

Some CLTs require you to specify directory paths. These are the full file system mount directory paths of a file in TeamSite, starting from the server's root directory. When you use a CLT, be sure to check whether it requires a directory path or a vpath.

For example, a file whose vpath is:

```
//IWSERVER/default/main/dev/WORKAREA/eng/htdocs/gifs/logo.gif
```

would have the directory path of:

```
Y:\default\main\dev\WORKAREA\eng\htdocs\gifs\logo.gif
```

Note: Vpaths can use forward or back slashes, whereas directory paths use back slashes.

Executing CLTs Remotely

If the `iw-home` directory is visible from the client machine you can run the CLTs from `iw-home/bin` without copying them into another location. The CLTs look for the libraries relative to the directory that contains the program file. For example, if the CLTs are in the default location (*iw-home*`\bin`), they require the NLS libraries to be located in *iw-home*`\lib\locale`.

To run CLTs from a remote system complete the following procedure:

1. Set the environmental variables as described in the previous section.

2. Map a drive to the `iw-home` directory on the system where the TeamSite server is installed.

3. Create the following string registry entry on your remote machine:

   \\HKLM\Software\Interwoven\TeamSite\iw-home = *iw-home*

   Where *iw-home* is the absolute path to the mapped `iw-home` directory. For example: `G:\Program Files\Interwoven\iw-home`.

Object IDs

Some CLTs require that you specify object IDs (`objids`). Objids are identifiers for each object (file, directory, workarea, staging area, edition, or tag) in the TeamSite system. To find the `objid` for an item, use the `iwattrib` command (see page 79). CLTs that require an `objid` also require further information about the `objid`, such as what type of object it describes or the area in which it resides.

Object IDs are 24-digit hexadecimal numbers, such as:

    0x0000225800000000000006bb

Class IDs

Each type of object has its own class ID. Class IDs are 8-digit hexadecimal numbers, whose values for each class are:

server	0x00000051
archive	0x00010010
branch	0x00010250
workarea	0x00010100
staging area	0x00010255
submit event	0x00002258
update event	0x0000225a
edition	0x00010020
file or directory	0x08001000

Specifying Comments

CLTs that use comments require them to be specified in one of two ways:

• Using a tag
• As a required field

For example, a CLT could require a comment to be specified using a tag:

```
iwsubmit [flags] [-c global_comment] [-i info_comment]
{vpath file_comment}+
```

Specify the comment using the `-c` tag. If you do not use the `-c` tag, the change comment will be read from `stdin`. After typing the comment, type `Ctrl-Z` to exit.

Note: If you do not specify a comment, you must still type `Ctrl-Z` to exit.

The second way of specifying a comment is as a required field:

```
iwmkwa //server/archive{[/branch]+} workarea comment base_ed
[ownername [groupname]]
```

This method requires you to type a comment in the Comment field. If you do not want to attach a comment, type two single quotes (`''`) in the Comment field. If the comment includes spaces, enclose it in a set of single quotes, for example, `'Please edit this file.'`

CLT Location

CLTs reside in the `bin` subdirectory of the directory returned by the `iwgethome` command. Note that not everything contained in this directory is a CLT.

Chapter 2

Administration Tools

This chapter describes the administration CLTs from the following groups:

- System Information Tools
- System Services Tools
- Backing Store Tools

The CLTs are grouped and summarized in the following sections. Each CLT is then presented in detail, arranged alphabetically.

System Information Tools

System Information CLTs return system-wide information or information about a file's location. They do not provide information about the data in a file (the Development Tool CLTs provide that type of information). CLTs in this category are:

CLT	Description	See...
iwancestor.exe	Returns the vpath of the common ancestor of the specified files.	page 25
iwgetelog.exe	Returns location of TeamSite events log.	page 42
iwgethome.exe	Returns location of TeamSite program files.	page 44
iwgetlocation.exe	Returns the locations of various TeamSite log and configuration files.	page 45
iwgetmount.exe	Returns location of TeamSite mount point	page 47
iwgetstore.exe	Returns location of TeamSite backing store.	page 48
iwgettrace.exe	Returns location of TeamSite trace logs.	page 49
iwrecentusers.exe	Displays a list of everyone who has used TeamSite since the last time the TeamSite server was started, and a timestamp of each user's most recent TeamSite operation.	page 55

INTERWOVEN

| iwstat.exe | Returns current system activity. | page 60 |
| iwversion.exe | Returns current TeamSite release information. | page 69 |

System Services Tools

System Services CLTs enable you to manipulate files or objects that affect the entire TeamSite system. Typical System Services tasks include, but are not limited to, editing configuration files, performing system backups, and starting the proxy server. CLTs in this category are:

CLT	Description	See...
iwabort.exe	Stops a current operation.	page 24
iwadduser.ipl	Adds a user to TeamSite role files.	page 26
iwchgrp.exe	Changes the group of a TeamSite workarea or branch.	page 27
iwconfig.exe	Reads or writes to TeamSite's main configuration file.	page 28
iwfreeze.exe	Freezes and unfreezes all system writes.	page 33
iwprefconv.exe	Copies existing homepage information from the iwprofiles directory to the entity database.	page 53
iwreset.exe	Rereads TeamSite configuration files.	page 56
iwrmuser.ipl	Removes a user from TeamSite.	page 57
iwsi.ipl	Collects system state data.	page 58
iwteamsite.exe	Installs or uninstalls the iwtock watchdog daemon.	page 64
iwtestcfg.exe	Returns the operation that will be performed upon a file at submission time.	page 65
iwuser	Enables you to manipulate TeamSite user information.	page 66

Backing Store Tools

Backing Store CLTs enable you to perform various tasks on the backing store. CLTs in this category are:

CLT	Description	See...
`iwconvert.exe`	Converts old-format (TeamSite 4.5.x and 5.0.x) backing stores to the new high-performance backing store format.	page 29
`iwconvertserver.exe`	Supports the conversion GUI by communicating with remote TeamSite servers and invoking `iwconvert` on behalf of the GUI. You must run the `iwconvertserver` process manually for the GUI to function properly.	page 32
`iwfsck.exe`	Diagnoses backing store problems.	page 35
`iwfsfix.exe`	Fixes problems found by `iwfsck`.	page 38
`iwfsshrink.exe`	Finds and removes duplicate data in the backing store.	page 40
`iwidmap.exe`	Change the mapping between the SID and the token. It can also be used to refresh the mapping when the same names are used, but the SID has changed.	page 50
`iwmigrate.exe`	Similar to `iwconvert` except that it accepts new-format backing stores as its source. It can be used to split a new-format backing store into multiple backing stores, or move the contents of a store to another location without losing the history of submit events for the editions.	page 51
`iwstoreadm.exe`	Enables you to create, activate, and deactivate new-format backing stores.	page 62

INTERWOVEN

iwabort.exe

Provides a method for terminating a long-running server operation. The command works on operations including submit, update, create branch, delete branch/workarea/edition, and freeze. Before issuing iwabort, use iwstat to obtain the identifier for the operation.

- Use with submit and update operations – There are occasionally some places in submit and update where they cannot be terminated by using iwabort. After issuing iwabort, you may end up with some but not all the files submitted or updated.

- Use with create and delete operations – The branch will not be created and may or may not be deleted.

Man Page Group:

System Services.

Usage:

iwabort *operation_id*

> *operation_id*   The operation_id displayed by iwstat.

Example:

Use iwstat to obtain the server status. Then, using the ID shown in the first column, abort the submit operation.

```
>iwstat

Status: Server running

ID          Thread  User    Duration  Operation
0x24caee4b  0x13    andre   560.006   SubmitFSE 0xef70087ced505b4004a9bc98
0x24caee58  0xf     -       0.000     RunFastVacuum
0x24caee57  0x15    chris   0.000     GetArchiveStatus

>iwabort 0x24caee4b
```

`iwancestor.exe`

Prints the vpath of the common ancestor of the specified files.

Man Page Group:

System Information.

Usage:

```
iwancestor [-i] vpath1 vpath2
```

`-i`	Prints the ID of the ancestor instead of the vpath.
`-h`	Prints usage message.
`-v`	Prints the version string.

INTERWOVEN

`iwadduser.ipl`

Adds a TeamSite user to the specified role files in `iw-home\conf\roles` and the entity database.

Man Page Group:

System services.

Usage:

iwadduser.ipl [-h|-V] -user <domain\user> -roles <roleList>

-h	Displays usage message.
-V	Verbose mode.
-user <domain\user>	The name of the user to be created.
-roles <roleList>	Comma-separated list of roles to which this user is assigned.

Example:

>**iwadduser -V -user factotum\jerome -roles master,admin**

Adds the user jerome to TeamSite master and admin roles files.

`iwchgrp.exe`

iwchgrp is chgrp for TeamSite for the Windows NT operating system. Because TeamSite retains the concept of group ownership of branches and workareas, iwchgrp allows you to change the group of a branch or workarea. To change the group of a branch or workarea, you must be the owner of that branch or workarea.

Man Page Group:

System Services.

Usage:

iwchgrp [-h|-v] *group path*

-h	Displays usage message.
-v	Displays version.
group	Specifies the new domain and group name.
path	Specifies the directory path (including the drive) to the workarea.

Examples:

>iwchgrp Administrators Y:\default\main\dev\Workarea\andre

changes the group of workarea andre on the dev branch to the local group of administrators.

>iwchgrp WEBSTER\Marketing Y:\default\main\dev\Workarea\andre

changes the group of workarea andre on the dev branch to the Marketing group in the WEBSTER domain.

INTERWOVEN

iwconfig.exe

A command line interface to TeamSite's main configuration file, `iw.cfg`. Allows you to read the configuration file, list the names of all the sections in the configuration file, or write to the configuration file. For information on configuring `iw.cfg`, see the "Configuring TeamSite" chapter of the *TeamSite Administration Guide*.

Man Page Group:

System Servicpes.

Usage:

iwconfig [*options*] *section varname* [*value*]

where `options` are:

-show	Displays the entire configuration file.
-l	Lists section names only.
-w	Writes value into /etc/iw.cfg (creates new section and variable if necessary).
-d	Delete a name, value pair from a specified section.
-h	Displays usage message.
-v	Displays version.
section	Section of iw.cfg to read or write to.
varname	Variable name within *section* to read or configure.
value	Value to assign to *varname* (use only with -w).

Example:

>iwconfig -l

returns:

```
main
iwcgi
iwproxy
iwproxy_remap
global_default_map
iwproxy_external_remap
iwserver
```

iwconvert.exe

The `iwconvert` CLT converts old-format (TeamSite 4.5.x and 5.0.x) backing stores to the new high-performance backing store format.

Complete the following steps to optimize the conversion process.

- Upgrade your source machine to TeamSite 4.5.1 Service Pack 2, or TeamSite 5.0.1 Service Pack 2 (or higher), with all available patches.

- Ensure you have the most recent version of the `iwconvert` tool.

  Updates to the `iwconvert` and `iwmigrate` tools shipped with TeamSite 5.5.1 will be available on the Interwoven support website. Before using either CLT, check the Interwoven support website (`http://support.interwoven.com`) to ensure you have the most recent version of each tool.

- Before running the `iwconvert` command, run `iwfsck -d` on your source backing store to prepare for conversion. The `iwfsck` CLT is described in the *TeamSite Command-Line Tools* manual.

Options

The following options are valid for the `iwconvert` command:

`-h`	Displays the usage message.
`-v`	Displays the version number.
`-b branch_vpath`	Location of the branch that contains the editions or workareas to be converted. If the `vpath` begins with `//hostname/` the branch is located on a remote TeamSite server.
	Note: Branches are converted recursively—all editions in subbranches under the specified branch are also converted unless the `-d` option is specified.

INTERWOVEN

`-c`	Cleans up the history information of a previously converted backing store. Requires that `-o` is also specified.
	This `iwconvert` step must be performed last, as a separate step, because it may have interbranch dependencies. Note that this action may safely be executed multiple times.
`-d`	Do *not* recursively convert subbranches.
`-s` *starting_edition*	Specifies the first edition in a range of editions to be converted for the specified branch (the default is `INITIAL`). Requires that `-b`, `-d`, `-o`, and `-m` are also specified.
`-e` *ending_edition*	Specifies the last edition in a range of editions to be converted for the specified branch (the default is the most recent edition). Requires that `-b`, `-d`, `-o`, and `-m` are also specified.
`-f`	Forces a full clone of each edition *without* the history of submit events for the editions.
	Use this option if there are gaps in the set of editions that you are converting, or if you do not want to save the submit history.
`-l` locale	Specifies the native locale of the backing store being converted and how non-ASCII metadata is interpreted. If this option is not specified it defaults to `LC_LOCAL`.
`-m` *iwmnt_mount_point*	Specifies the mount point for the existing (source) `iwserver` installation. Required with `-b`, `-r`, and `-w`.
`-n` *old_backing_location*	Use direct access to the old backing store for faster conversions. Can be used with `-b`, `-r`, or `-w`.
	If this option is not specified, `iwconvert` will run slowly due to calls to `sci_GetPredecessors()`.
`-o` *new_backing_location*	Location of the new backing store. This must be a path to the store root. For example, the store named `default` is specified by: `/local/iw-store/default`.

-w *workarea_name*	Converts the specified workarea. Requires that –b, -o, and –m are also specified.
-x	Increases the verbosity level. Maximum verbosity is level 3, expressed as -x -x -x.
Ctrl+c	Stops iwconvert at the end of the edition currently being converted. When you restart the conversion, iwconvert ignores the editions in the branch that have already been converted and converts the remaining editions.

Usage Summary

- Convert editions:
  ```
  iwconvert –o new_backing_store_location -m iwmnt_mount_point
  [-n old_backing_store_location] -b branch_vpath [-d [-s
  starting_edition] [-e ending_edition] ]
  ```

- Convert a workarea:
  ```
  iwconvert –o new_backing_store_location -m iwmnt_mount_point
  [-n old_backing_store_location] -b branch_vpath -w workarea_name
  ```

- Clean up history:
  ```
  iwconvert –o new_backing_store_location -c
  ```

Example
```
iwconvert -m f:/ -o d:/iw-store/Safari -n C:\iw-store\default
-b //bgunn/default/main/www
```

INTERWOVEN

`iwconvertserver.exe`

The `iwconvertserver` process supports the backing store conversion GUI by communicating with remote TeamSite servers and invoking `iwconvert` on behalf of the GUI. You must run the `iwconvertserver` process manually for the GUI to function properly.

Usage
```
iwconvertserver -S [-h|-v|-V level|-is|-us]
```

Server Mode Options

`-S`	Server mode
`-is`	Install NT Service
`-us`	Uninstall NT Service

General Options

`-h`	Print this message
`-v`	Print version
`-V level`	Specify verbose logging level (1-5, default is 1)

Procedure

1. Change to the `iw-home/bin` directory, for example:

   >**cd C:\Program Files\Interwoven\TeamSite\bin**

2. Either:
   - Run the `iwconvertserver` utility directly from the command line:
     iwconvertserver -S
   - Install it as a service, and then start the service using the Service Control Manager:
     iwconvertserver -is

`iwfreeze.exe`

Freezes and unfreezes all system writes. Unlike a full server stop, users can still log into the system and read data.

To find out whether the TeamSite server is currently frozen, use the `iwstat` (page 60) command-line tool.

Man Page Group:

System Services.

Usage:

`iwfreeze [-h|-v|+N|-j|--]`

-h	Displays usage message.
-v	Displays version.
+N	Freeze for N seconds.
-j	Freeze or unfreeze only batch jobs, not the entire archive. As currently supported, batch jobs result from a Delete area operation and accomplish the following: clean up destroyed workareas, editions, and branches.
--	Unfreeze archive immediately.

INTERWOVEN

Examples:

>iwfreeze +30

returns:

```
Freeze archive default for 30 seconds...
```

and freezes the TeamSite archive for thirty seconds.

>iwfreeze --

returns:

```
Archive default unfrozen.
```

and unfreezes the archive.

Notes:

When iwfreeze is issued, the system performs the following activities:

- blocks new operations
- completes all current operations
- flushes the cache
- issues a message telling users an iwfreeze has occurred
- returns write operations with a failure status message
- performs read-only operations

`iwfsck.exe`

Diagnoses backing store problems and allows repair of some of the problems found.

Man Page Group:

Backing Store.

Usage:

```
iwfsck [-h] [-v] [-xxx] [-l] [-y] [-b path] [vpath] [-scan] [-noscan]
[-nofmtchk] [-z] [-d [-f] [-s] [-m] [-p] [-r]] [-o file] [-e file]
[-t file] [-u file]
```

-h	Displays usage message.
-v	Displays version.
-x	Requests extra output and increments verbosity level. Prints additional information about what iwfsck is doing as it operates. Each x increments the verbosity level by 1. The highest level of verbosity is level 3 (-xxx). In the higher levels of verbosity, an extremely large quantity of output may be produced.
-c *size*	Sets internal cache size (for debugging iwfsck only).
-w *size*	Sets nominal fastwalk memory limit (default is unlimited, size greater than 192 recommended if this option is used).
-l	Prints output as HTML. This option is used by the iwfsckcgi.cgi program.
-y	Repairs damaged files while running. In this mode, damaged files are deleted while iwfsck is running. The TeamSite server must be down when specifying this option. If the TeamSite server is running when this option is specified, a warning displays and this option is ignored.
-b *path*	Uses *path* as the backing store location. The default is the configured backing store location returned by iwgetstore for the TeamSite server.
-nofmtchk	Inhibits validation of FORMAT files.

INTERWOVEN

`-report`	Forces the reporting of repair actions even if the store is in use by another process.
`-scan`	Performs a preliminary point scan and validation pass (warning, can be resource intensive).
`-noscan`	Inhibits preliminary point scan and validation pass.
`-z`	Checks events in branches.
`-d`	Checks directories and files in addition to the normal checking of branches and areas. All directories and files from the *vpath* are walked. If a *vpath* is not specified on the command line, the walk begins at the / vpath.

The following options are only allowed when `-d` is specified:

`-unpub`	Checks directories and files in unpublished editions. This is very memory intensive and is not recommended. If you do use this option, using `-s` also will reduce memory usage, but will be very slow.
`-f`	Provides a fast reference check (not allowed with `-p`, `-m`, or `-s`). All references from the root are walked aggressively looking for missing references. If a missing reference is found, that part of the tree is marked *suspect*, and a more expensive walk with vpaths is done on that part of the tree to determine the directories and files affected by the problem.
`-s`	Provides a stack walk, which is slower but uses less memory than the default (not allowed with `-f`). This mode uses the least amount of memory, but it is the least efficient for walking the entire tree of files and directories from the root.
`-m`	Checks ModLists for directories. A ModList is a data structure that is a shadow tree to the directory structure within a workarea. This shadow tree allows the modified files within a workarea to be

	determined quickly without having to traverse every file and directory within a workarea.
-p	Checks protopaths. A protopaths is a data structure that allows file names and history information to be determined without the expense of walking up to the root of an area through directories; however, it can be expensive.
-r	Checks parents. Parents and anti-parents are the reference counting mechanism used by the TeamSite server. If zero parents are found for a file, it indicates a problem. It can be expensive.

The following options specify where output goes (note that stdout and stderr may be redirected in the normal way in a command line shell and that -o and -e are provided to enable redirection when shell redirection is not available):

-o *file*	Specifies output file for server startup information.
-e *file*	Specifies the file to write error messages to.
-t *file*	Specifies the file to write reports to.
-u *file*	Specifies the summary file.
vpath	Specifies the starting vpath to walk directories when -d is used.

Examples:

To check areas and branches, issue the command:

>**iwfsck**

To check directories and files in addition to branches and areas, issue the command:

>**iwfsck -d**

Use the following command to check protopaths and parents in addition to branches, areas, directories, and files. This command can be very resource intensive.

>**iwfsck -d -p -r**

INTERWOVEN

`iwfsfix.exe`

The `iwfsfix` command repairs certain problems. If `iwfsck` (see page 35) finds problems when the backing store is diagnosed, it outputs lines in the format:

```
FIX iwfsfix repair args
```

The repairs and their arguments are shown below. To perform necessary repairs, copy the `FIX` line issued by `iwfsck` and paste it on the command line, with the word `FIX` removed.

There are also repairs for ModLists that must be performed when the TeamSite server is running. On Windows platforms, the ModList repair lines are in the format:

```
FIX echo x > junkfile; del /f junkfile
```

`junkfile` is a uniquely named file that is created and removed from an affected directory.

The repairs that can be performed with `iwfsfix` are:

- `delete_tag branch_id tag_id`

  Removes the reference to a tag (lock) from a branch. This is done when the tag point itself is missing.

- `delete_tag_and_point branch_id tag_id`

  Deletes the reference to a tag (lock) from a branch and removes the tag point itself. This is done when a tag duplicates or conflicts with another tag within a branch.

- `delete_direntry directory_id diritems_index filename`

  Deletes the directory entry for a damaged or missing file.

- `replace_direntry directory_id diritems_index filename new_standin_id`

  Repairs a directory entry to point to a correct standin ID.

- `delete_area area_id`

  Deletes the point for an area.

- `delete_area_from_branch branch_id area_id workarea | edition`

  Deletes the reference to an area (workarea or edition) from a branch. This cannot be done on a staging area because a branch by definition always contains a staging area.

- `null_previous` *point_id*

  Sets to null (-1) the PreviousPoint reference within a point. This is done when the PreviousPoint reference for a point is incorrect.

- `clone_diritems` *directory_id diritems_index new_gen_id new_dot_dot*

  Clones a set of directory items within a directory to create a new set. This is done when a set of directory items is shared between areas, but it should not be shared.

INTERWOVEN

iwfsshrink.exe

Invokes a batch job in the server to find and remove duplicate file contents in the backing store. This command is used to improve space utilization in an existing backing store. It can be used as a maintenance tool on a regular, perhaps quarterly, basis. This operation results in no user-visible changes to the TeamSite virtual file system; for example, file histories are unchanged.

Man Page Group:

Backing Store.

Usage:

```
iwfsshrink [-h|-v] [run|pause|abort|status]
```

-h	Displays usage message.
-v	Displays version.
run	Starts the iwfsshrink process.
pause	Temporarily stops the iwfsshrink process. It can be restarted with the run option. Because iwfsshrink takes a long time to run and it increases system load, you may want to start it during off-hours. When activity increases, you can pause it until the next period of inactivity.
abort	Terminates the iwfsshrink process.
status	Shows information about the iwfsshrink process.

Examples:

Issuing iwfsshrink status when iwfsshrink has finished running, results in status similar to:

```
Not currently running.
Last started Mon Jun 26 15:47:53 2000
Last completed Tue Jun 27 00:40:04 2000
Files examined: 317974
Bytes examined: 75936814830
Files found to be duplicates: 233430
Files converted: 198352
Bytes removed: 23455046531
```

Issuing the `iwfsshrink run` command, followed by the `iwfsshrink status` command, results in status similar to:

```
Current run started Tue Jul 11 14:19:20 2000
Not yet completed
In phase 1
Files examined: 980
Bytes examined: 67053122
Files found to be duplicates: 0
Files converted: 0
Bytes removed: 0
```

Issuing the `iwfsshrink pause` command, followed by the `iwfsshrink status` command, results in status similar to:

```
Current run started Tue Jul 11 14:19:20 2000
Paused
In phase 1
Files examined: 5694
Bytes examined: 1191581411
Files found to be duplicates: 0
Files converted: 0
Bytes removed: 0
```

INTERWOVEN

iwgetelog.exe

Returns the contents of /etc/defaultiwelog (the TeamSite events log) or a non-zero error code. The TeamSite events log includes the time an operation was performed, the username and role of the user, the type of operation, the area or file affected, and additional information according to the type of operation performed (for example, objids and comments attached to files and operations).

Man Page Group:

System Information.

Usage:

iwgetelog [-h|-o|-v]

-h	Displays usage message.
-v	Displays version.
-o	Returns original factory setting value.

Example:

>iwgetelog

returns:

C:\Program Files\Interwoven\TeamSite\local\logs\iwevents.log

iwgetfilejobs.exe

Returns a list of associated workflow job and task IDs for a file.

Man Page Group:

System Information.

Usage:

```
iwgetfilejobs [-h] [-v] [-s servername] path
```

-h	Displays usage message.
-v	Displays version.
-s	Use servername as TeamSite server.
path	Path to look up (including the file name).

Example:

>iwgetfilejobs -s factotum \default\main\WORKAREA\connor\foo.txt

returns:

```
job=125998      task=125999
```

You can use the returned job or task ID with a command like `iwgetwfobj` as described on page 102.

INTERWOVEN

`iwgethome.exe`

Returns the location of the TeamSite program files (the contents of `/etc/defaultiwhome`).

Man Page Group:

System Information.

Usage:

```
iwgethome [-h|-v]
```

`-h`	Displays usage message.
`-v`	Displays version.

Example:

>iwgethome

returns

```
C:\Program Files\Interwoven\TeamSite
```

`iwgetlocation.exe`

Returns the location of various TeamSite configuration or log files.

Man Page Group:

System Information.

Usage:

```
iwgetlocation [-h|-v] [-a|-l|-c configfilekey|-g logfilekey|filekey
[filekey...]]
```

-h	Displays usage message.
-v	Displays version.
-a	Returns all locations of all configuration and log files.
-l	Lists all filekeys.
-c *configfilekey*	Specifies a configuration file.
-g *logfilekey*	Specifies a log file.
filekey	Specifies a particular configuration or log file.

Examples:

>iwgetlocation -l

returns:

```
iwhome
iwbin
iwconfig
iwstore
iwmount
iwcgimount
iwroles
iwlogs
iwconfigs
iweventlog
iwtracelog
iwdeploylog
```

INTERWOVEN

>iwgetlocation -a

returns:

```
iwhome=C:\Program Files\Interwoven\TeamSite
iwbin=C:\Program Files\Interwoven\TeamSite\bin
iwconfig=C:\Program Files\Interwoven\TeamSite\etc\iw.cfg
iwstore=C:\iw-store
iwmount=Y:
iwcgimount=Y:
iwroles=C:\Program Files\Interwoven\TeamSite\conf\roles
iwlogs=C:\Program Files\Interwoven\TeamSite\local\logs
iwconfigs=C:\Program Files\Interwoven\TeamSite\local\config
iweventlog=C:\Program Files\Interwoven\TeamSite\local\logs\iwevents.log
iwtracelog=C:\Program Files\Interwoven\TeamSite\local\logs\iwtrace.log
iwdeploylog=C:\Program Files\Interwoven\TeamSite\local\logs\iwdeploy.log
```

>iwgetlocation iwmount

returns:

```
Y:
```

>iwgetlocation -c iwconfig

returns:

```
C:\Program Files\Interwoven\TeamSite\etc\iw.cfg
```

>iwgetlocation iwmount iwconfig iweventlog

returns:

```
Y:
C:\Program Files\Interwoven\TeamSite\etc\iw.cfg
C:\Program Files\Interwoven\TeamSite\local\logs\iwevents.log
```

`iwgetmount.exe`

Returns the location of TeamSite mount point (the contents of `/etc/defaultiwmount`).

Man Page Group:

System Information.

Usage:

`iwgetmount [-h|-o|-v]`

Returns the contents of `/etc/defaultiwmount` or a non-zero error code.

-h	Displays usage message.
-v	Displays version.
-o	Returns original factory setting value.

Example:

>iwgetmount

returns:

`Y:`

`iwgetstore.exe`

Returns the contents of `/etc/defaultiwstore` (the location of the TeamSite backing store) or a non-zero error code.

Man Page Group:

System Information.

Usage:

`iwgetstore [-h|-o|-v]`

`-h`	Displays usage message.
`-v`	Displays version.
`-o`	Returns original factory setting value.

Example:

>iwgetstore

returns:

`C:\iw-store`

`iwgettrace.exe`

Returns the contents of `/etc/defaultiwtrace` (the location of the TeamSite trace logs) or a non-zero error code. The trace log contains debug information for TeamSite.

Man Page Group:

System Information.

Usage:

`iwgettrace [-h|-o|-v]`

`-h`	Displays usage message.
`-v`	Displays version.
`-o`	Returns original factory setting value.

Example:

>iwgettrace

returns:

`C:\Program Files\Interwoven\TeamSite\local\logs\iwtrace.log`

INTERWOVEN

iwidmap.exe

The `iwidmap` CLT changes the mapping between the SID and the token. It can also be used to refresh the mapping when the same names are used, and the SID has changed. For more information about SID mapping, see "SID Changes to the TeamSite Backing Store" in the *TeamSite Administration Guide*.

Usage

```
iwidmap [-v] [-h] (-u | -g) [-a][-c <user1> <user2>] [-x <file> | -i <file>]
backing-store
```

`-v`	Displays the version of this program.
`-h`	Displays the usage message.
`-u`	Update `userid` mapping.
`-g`	Update `groupid` mapping.
`-a`	Update all entries in the ID map.
`-c <user> <user2>`	Update user1 to user2.
`-x <file>`	Extract to file.
`-i <file>`	Import from file.
`backing-store`	Location of the backing store.

Example

```
iwidmap -u -c jgarcia rhunter c:\iw-store\NewReleases
```

iwmigrate.exe

The iwmigrate CLT is similar to iwconvert except that it accepts new-format backing stores as its source. It can be used to split a single new-format backing store into multiple backing stores, or to move the contents of a store to another location without losing the history of submit events for the editions.

Note: Updates to the iwconvert and iwmigrate tools shipped with TeamSite 5.5.1 will be available on the Interwoven support website. Before using either CLT, check the Interwoven support website (http://support.interwoven.com) to ensure you have the most recent version of each tool.

Usage

```
iwmigrate [-h] [-v] [-x] [-m mount_location] -o new_backing_location
[-b branch_vpath] [-s starting_ed] [-e ending_ed] [-n old_backing_location]
[-w workarea_name] [-c] [-d] [-f] [-l]
```

-h	Display this message.
-v	Display version number.
-b branch	Specify source branch for migration.
-x	Increase verbosity level. Maximum verbosity is level 3, expressed as -x -x -x.
-m mount_location	Specify mount location of backing store, for example: F:\
-o new_backing_location	Specify new backing store location.
-n old_backing_location	Specify old backing store location.
-d	Do *not* recursively convert subbranches
-s starting_ed	Specify starting edition for migration. Default is the INITIAL edition (this option can only be used with -d)
-e ending_ed	Specify ending edition for migration. Default is the most recent edition.
-f	Full clone of every edition (does *not* preserve history).

INTERWOVEN

`-r`	Clean up history information (use on the final pass).
`-l locale`	Specify the native locale of the backing store being migrated (if different from `LC_LOCAL` for this system).

Example

`iwmigrate -m F:\Safari -o d:\iw-store\safari_on_line -b`

`iwprefconv.exe`

The SetHomePage functionality in the WebDesk Pro GUI now stores the homepage information in the entity database instead of the `iwprofiles` directory.

If you are upgrading to TeamSite 5.5.1, you must run the `iwprefconv` CLT once to copy any existing homepage information from the `iwprofiles` directory (`iw-home/local/iwprofiles`) to the entity database (`iw-home/local/entities/data`).

If you are installing TeamSite 5.5.1 without upgrading from a previous TeamSite release, the homepage information is stored in the entity database by default.

To run the `iwprefconv` CLT, `iwserver` must be running.

Man Page Group:

System Services.

Usage:

`iwprefconv.exe`

(The CLT does not accept any parameters.)

INTERWOVEN

iwproxy.exe

Used to invoke and debug the proxy server. The proxy server is automatically invoked at startup time.

Man Page Group:

System Services.

Usage:

```
iwproxy [-v|-h|-d|-x]
```

-h	Displays usage message.
-v	Displays version.
-d	Debug mode (outputs client & server headers).
-x	Extended debug mode (verbose output).

Example:

>iwproxy -d

returns:

```
Using tickets for authentication
Using newfangled blobs
[iwproxy] max_connections (default): 100
[iwproxy: using gethostname() for local hostname: factotum]
[iwproxy: changed local hostname to result of gethostbyname(factotum):
    factotum.interwoven.com]
[iwproxy] listening at http://factotum.interwoven.com:1080
[iwproxy] using customer web server at http://factotum.interwoven.com:81
[iwproxy] using servlet engine at http://localhost:8080/iw-ts
[iwproxy] using iwwebd at http://fully.qualified.hostname.com:80
[external remap configuration]
[name not set]
key: 0
docroot: [no docroot specified]
[local remap configuration]
/
key: 0
docroot: [no docroot specified]
```

`iwrecentusers.exe`

Displays a list of everyone who has used TeamSite since the last time the TeamSite server was started. The list also contains a timestamp of each user's most recent TeamSite operation.

Man Page Group:

System Information.

Usage:

```
iwrecentusers [-h|-v|-Gn]
```

-h	Displays usage message.
-v	Displays version information.
-G0	Output is tab-separated. If a field is longer than the tab, then alignment will be erratic.
-G1	Output is neatly formatted (default). Field width is adjusted for the longest value in the column and is aligned.

INTERWOVEN

iwreset.exe

Tells the TeamSite server to reread the TeamSite configuration files (`iw.cfg`, `submit.cfg`, `autoprivate.cfg`, and the roles files) and resets `iwserver` based on the values in the configuration files.

Man Page Group:

System Services.

Usage:

```
iwreset [-a|-h|-ui|-v]
```

`-a`	Reread configuration and then restart services (as if the `-ui` option had been issued).
`-h`	Displays usage message.
`-ui`	Restart (stopping if necessary) the following `ui` services: `iwopenapi`, `iwebd`, `iwproxy`, `iwservletd`.
`-v`	Displays version.

Example:

>iwreset

Tells the TeamSite server to read the TeamSite configuration files and reset `iwserver`.

Notes:

Exits with 0 on success, non-zero on failure.

All except the following `iw.cfg` settings are reset when `iwreset` executes. You must restart TeamSite for the following settings to reset.

- Cache size (set in [`iwserver`])
- File system thread count (set in [`iwserver`])
- File system active area cache (set in [`iwserver`])
- File locations (set in [`locations`])

`iwrmuser.ipl`

Removes a TeamSite user from the specified role files in `iw-home\conf\roles` and the entity database.

Man Page Group:

System Services.

Usage:

```
iwrmuser.ipl [-h|-V] <domain\user>
```

-h	Displays usage message.
-V	Verbose mode.
domain\user	The user to remove.

Example:

>**iwrmuser.ipl factotum\jerome**

Removes the user jerome from TeamSite. Note that unlike iwadduser, you *cannot* selectively remove users from specific role files.

INTERWOVEN

`iwsi.ipl`

Available only if the high availability version of TeamSite is installed. Collects TeamSite state data about the local environment, including core and configuration files. Files collected are not deleted from their original locations. Data is stored and compressed in an archive file named `si.`*`timestamp`* in either *`iw-home`* or a specified alternate location. The command exits after executing; it does not run as a daemon.

Man Page Group:

System Services.

Usage:

`iwsi.ipl [-h|-v] [`*`file`*`]`

`-h`	Displays usage message.
`-v`	Displays version. Has highest precedence if multiple options are specified.
`file`	Specifies the full directory path of an alternate archive location. Each archive file name is given a timestamp suffix to distinguish it from others with similar names.

Exit Status:

The following exit values are returned:

`O`	Successful options processing
`-1`	Invalid options; could not properly set the path; could not find all the necessary binaries
`>1`	Other processes returned an error

Notes:

The value of `timestamp` in the archive file name has the format `yymmdd.hhmmss`.

The archive contains the following files:

Log Files:

```
iwevents.log
iwserver.log
iwtrace.log
iwdeploy.log
```

Configuration Files:

```
iw.cfg
$IW_HOME/local/config/submit.cfg
$IW_HOME/local/config/autoprivate.cfg
```

Core Files:

```
system.dmp
memory.dmp
```

Server and Process Output Files:

```
iwserver
information<processid>
```
(a file containing output from `iwversion` command)

INTERWOVEN

iwstat.exe

Returns current system activity and displays server state information about whether the server is running, frozen (with iwfreeze), flushing, or out of space.

Man Page Group:

System Information.

Usage:

iwstat

-h	Displays usage message.
-v	Displays version
-c	Displays cache statistics.
-G0	Output is tab-separated.
-G1	Output is neatly formatted (default).

Example:

While a Submit operation is underway,

>iwstat

returns:

```
ID      Thread User           Duration Operation
0x8738 0xe1    WEBSTER\andre 0.901     SubmitFSE 0x00002100000000000002134
0x873a 0xfa    Administrator 0.000     GetArchiveStatus
```

While the TeamSite server is frozen as the result of `iwfreeze`,

```
>iwstat
```

returns:

```
*** SERVER FROZEN: 55 SECONDS REMAINING ***
ID       Thread  User    Duration   Operation
0x9d     0xf     root    0.000      GetArchiveStatus
```

The following output from `iwstat` was obtained while the server was running.

```
Status: Server running

ID       Thread  User    Duration   Operation
0x3f4e   0x17    mroot   0.000      GetArchiveStatus

Minutes   Thruput  Avg op  Load
       1       27  0.0189  0.51
      15       14  0.0219  0.30
      29        9  0.0233  0.20
```

Explanation: In the most recent complete one-minute time duration, the TeamSite server handled 27 operations per second. Each operation took an average of 0.0189 seconds per operation (19 ms/operation). The average load over this one minute was 51 percent. By comparison, a single-CPU server is fully loaded at 100 percent. A two-cpu server is fully loaded at 200 percent load, etc. Similarly, for a recent complete 15-minute duration, TeamSite handled 14 operations per second, with an average operation time of 22 ms. Over this time, the server was occupied at 30 percent load.

Comparing the short and long duration figures shows the load trends. For example, a high one-minute load, compared with a lower 60-minute load, tells you that there was a burst of activity. You can also observe that this is not a fast server, because 27 operations per second at 50 percent load is not much throughput. A fast server with fast disks can achieve operation throughput in the 200-500 operations per second range, with operation times in the 1-5 ms/operation range.

Finally, a load that is near or exceeds the number of CPUs on the server over a sustained number of hours may indicate an overloaded server. It may also indicate a need to obtain a more powerful server or a faster backing store volume.

INTERWOVEN

iwstoreadm.exe

Backing store administration involves creating, activating, and deactivating backing stores by using the `iwstoreadm` CLT. When the `iwstoreadm -a storename` command is issued, the following sequence is triggered:

- The `iw.cfg` file is checked to see if a `store_directory` or `store_comment` entry exists, if it does, their definitions are used to create the backing store. If these entries do not exist:

- The backing store directory is automatically created in `C:\iw-store\` and populated with a minimal backing store containing a branch named `main`. The store remains active until explicitly deactivated by using `iwstoreadm -d` (even if the server is stopped and restarted).

  Deactivating a store does *not* delete it. A store can be deactivated, moved to a new location, and reactivated using original name though it will be assigned a new store ID.

iwstoreadm.exe Options

The following options are valid for the `iwstoreadm` command:

`-a store_name`	Activates an existing store, or creates and activates a new store. You must have an entry in the Master role file (*iw-home*\conf\roles\master.uid) to execute this option.
`-d store_name`	Deactivates an existing store. You must have an entry in the Master role file (*iw-home*\conf\roles\master.uid) to execute this option.
`-h`	Displays the usage message.
`-l`	Lists active stores.

Usage
`iwstoreadm [-l] [-a store_name] [-d store_name]`

Example
`>iwstoreadm -l`

Displays the active backing stores:

```
Name       Store Directory      ID          Comment
----       ---------------      --          -------
default    C:\iw-store\default  0x64
store2     C:\iw-store\store2   0x65
```

Example

>**iwstoreadm -a c:\temp\store1 Store1 "Test store for demo"**

Note that the comment must be in quotes if it contains spaces.

The `store1` directory and `Store1` backing store are created. The system confirms that the newly created backing store is activated.

INTERWOVEN

iwteamsite.exe

Available only if the high availability version of TeamSite is installed. Displays information about the TeamSite `iwtock` watchdog daemon and lets you install or uninstall `iwtock` as a service.

Man Page Group:

System Services.

Usage:

```
iwteamsite [-h|-v|-install|-uninstall|-d]
```

-h	Displays usage message.
-v	Displays version. Has highest precedence if multiple options are specified.
-install	Installs `iwtock` as a service. Fails if `iwtock` is already installed as a service.
-uninstall	Uninstalls `iwtock` as a service. Fails if `iwtock` is not already installed as a service.
-d	Execute `iwtock` as a command line process (useful for debugging).

Exit Status:

The following exit values are returned:

0	Successful options processing.
-1	Invalid options; could not properly set the path; could not find all the necessary binaries.
>1	Other processes returned an error.
127	Exit was requested.

Notes:

When you execute `iwteamsite` with an option, it exits upon completion (as opposed to running as a daemon).

See *TeamSite Administration Guide* for more information about the high availability version of TeamSite.

`iwtestcfg.exe`

Returns the submit filtering operation that will be performed upon a file at submission time. For more information about submit filtering, see the *TeamSite Administration Guide*.

Man Page Group:

System Services.

Usage:

```
iwtestcfg [-h|-v] vpath
```

-h	Displays usage message.
-v	Displays version.
vpath	Specifies a vpath to a file (see page 15).

Example:

>iwtestcfg /default/main/WORKAREA/andre/cgi/test.sh

returns:

```
Matched area "^/default/main/WORKAREA/.*$"
Matched fse ".*\.sh$"
Actions to do are:
omask=0111
```

which means that TeamSite is configured to set execute permissions on the specified file at the time that the file is submitted.

I N T E R W O V E N

iwuser

Enables you to manipulate TeamSite user information.

Man Page Group:

System Information.

Usage:

iwuser *action ident* [options]

where *action* is one of:

-create or -c	Creates a new entry.
-delete or -d	Deletes an existing entry.
-modify or -m	Modifies an existing entry.
-query or -q	Queries an entry only (default).

where *ident* is one of:

[-name] nm	Works with entry whose name is nm.
-id eid	Works with entry whose ID is eid.
-group	Entry refers to a group.
-loc el	Works with entry whose locator is el.
-session s	Extracts user name from session string s.
-user	Entry refers to a user (default).

ident Notes:

- -user and -group are mutually exclusive.
- Windows does not distinguish between -user and -group.
- -session is mutually exclusive with -name and implies -user.
- The syntax of an entry ID is *T*::*SID*: where : characters are significant. *T* should be U for a user entry or G for a group. *SID* is a Windows SID (in hex).

Options available for all actions:

`-a`	Automatically creates user or group if it does not already exist.
`-h`	Displays this help information and exit.
`-v`	Displays version information and exit.
`-D`	Use the local user database directly.
`-F path`	Specifies the path of the local user database.

Options available for create actions only:

`-l nm[,nm...]`	Sets the list of aliases to `nm,nm...`
`-g nm[,nm...]`	Sets the list of related groups to `nm,nm...`

Options available for modify actions only:

`-l nm[,nm...]`	Sets the list of aliases to `nm,nm...`
`-la nm`	Adds name `nm` to list of aliases.
`-ld nm`	Deletes name `nm` from list of aliases.
`-g nm[,n...]`	Sets or replaces the list of related groups.
`-ga nm`	Appends name `nm` to list of related groups.
`-gd nm`	Deletes name `nm` from list of related groups.
`-gi nm posnm`	Inserts `nm` before `posnm` in list of related groups.
`-gp nm`	Prefixes name `nm` to list of related groups.

Modify option notes:

- `-l` and `-g` *cannot* be specified more than once; all others in this list may.
- Options are processed in the order they are specified.
- If `nm` contains a `:` then it is assumed to be an entity locator, otherwise it is treated as a user or group name as appropriate.

I N T E R W O V E N

Options available for all actions *except* delete:

`-n`	Suppress inheritance from relatives.
`-qa`	Show all user/group information (default).
`-qg`	Print names of related groups (one per line).
`-qi`	Print entry ID.
`-ql`	Print aliases.
`-qn`	Print local name alias.
`-qr nm`	Print RELATED if group `nm` is related, print UNRELATED if not.

Notes (applies to all actions *except* delete):

- One or more of these may be specified. If more than one is present, the results are printed in the same order, separated with blank lines.
- Queries are done *after* create or modify actions.

`iwversion.exe`

Returns the current TeamSite release information. It is often used as a test to see if the TeamSite server is running. Executing `iwversion` makes a sciface RPC call to `iwserver` and reads the version string from `iwserver`.

Man Page Group:

System Information.

Usage:

```
iwversion [-h|-v]
```

`-h`	Displays usage message.
`-v`	Displays the version string.

Note: When `iwserver` is running, `iwversion` displays the same result as `iwserver -v`. `iwversion -v` displays the version string that is embedded into the `iwversion` CLT. If you have installed patches, the two version strings may not be the same.

Example:

>iwversion

returns:

```
iwserver: 5.0.0 Build 420 Interwoven 20000812
```

Chapter 3

Development Tools

This chapter describes the development CLTs from the following groups:

- General Development Tools
- Deployment Tools
- Branch Operation Tools
- Edition Operation Tools
- Workarea Operation Tools
- Version Management Tools
- Workflow and Job Tools

The CLTs are grouped and summarized in the following sections. Each CLT is then presented in detail, arranged alphabetically.

General Development Tools

General Development CLTs enable you to perform operations on, or retrieve information about, the data in a specific file or TeamSite object. CLTs in this category are as follows:

CLT	Description	See...
iwattrib.exe	Returns metadata information on any object in the TeamSite server, including all branches, workareas, editions, staging areas, files, directories, and symlinks.	page 79
iwckrole.exe	Checks whether or not a user can log in with a particular role.	page 91

INTERWOVEN

iwdecode.exe	Decodes HTML-encoded %xx lines to ASCII lines.	page 94
iwencode.exe	Encodes ASCII lines to HTML-encoded %xx lines.	page 96
iwextattr.exe	Sets extended attributes on a file.	page 100
iwlist.exe	Returns a list of all areas contained by a parent areas. For example, iwlist returns all editions, workareas, and branches contained on a parent branch.	page 113
iwmerge.ipl	Merges files from one area into another.	page 119
iwrename.exe	Renames a file, directory, workarea, edition, or branch.	page 128
iwvpath.exe	Prints all or parts of the version path of a specified object.	page 150

Deployment Tools

The Deployment CLT enables you to deploy a TeamSite edition to a production server. The CLT in this category is:

CLT	Description	See...
iwdeploy.exe	Deploys website content to the production server.	page 95

Branch Operation Tools

Branch Operation CLTs enable you to create and manipulate TeamSite branches. CLTs in this category are as follows:

CLT	Description	See...
iwmkbr.exe	Creates a new TeamSite branch.	page 120
iwrmbr.exe	Deletes a TeamSite branch and all contained areas (workareas, staging areas, and editions).	page 133

Edition Operation Tools

Edition Operation CLTs enable you to create and manipulate TeamSite editions. CLTs in this category are as follows:

CLT	Description	See...
iwlasted.exe	Returns the name of the last published edition on a branch.	page 112
iwnexted.exe	Returns the name that would be autogenerated for the next edition on a branch.	page 122
iwpublish.exe	Publishes the staging area.	page 124
iwrmed.exe	Deletes an edition.	page 134

Workarea Operation Tools

Workarea Operation CLTs enable you to create and manipulate TeamSite workareas. CLTs in this category are as follows:

CLT	Description	See...
iwevents.exe	Returns a history of submissions or updates of a workarea.	page 97
iwlistlocks.exe	Lists the locks and assignments in a workarea or branch.	page 115
iwlistmod.exe	Lists all modified files and directories in a specified area.	page 116
iwlock.exe	Locks any file or directory in a specified area.	page 117
iwlockinfo.exe	Provides detailed information on any lock.	page 118
iwmkwa.exe	Creates a new TeamSite workarea.	page 121
iwprv.exe	Allows users to identify whether any file or directory in any workarea is marked private and automatically mark any file as either private or public.	page 123
iwrmwa.exe	Deletes a TeamSite workarea.	page 139
iwsubmit.exe	Submits a file, directory, or entire workarea to the staging area.	page 143

| iwunlock.exe | Unlocks any file or directory. | page 148 |
| iwupdate.exe | Updates a TeamSite workarea with the version of a file or directory in the staging area. | page 149 |

Version Management Tools

Version Management CLTs return information about file revisions and versions, and enable you to manipulate specific versions of files and directories. CLTs in this category are as follows:

CLT	Description	See...
iwcat.exe	Displays any version of a (text) file in TeamSite.	page 90
iwcmp.exe	Compares any two TeamSite areas and returns a list of differences.	page 92
iwrevert.exe	Revert to a previous version of a file.	page 130
iwrlog.exe	Shows a revision log for a TeamSite file.	page 131

Workflow and Job Tools

Workflow and Job CLTs enable you to instantiate or destroy jobs, view the state of any object in the workflow system, add files to a particular task, and otherwise interact with running jobs. These tasks are typically performed by Administrators or Master users. CLTs in this category are as follows:

CLT	Description	See...
iwaddtaskfile.exe	Adds a file to a job task that is part of a job instance already created on the TeamSite server.	page 76
iwaddtaskgroup.exe	Adds groups to group tasks.	page 77
iwaddtaskuser.exe	Adds users to tasks.	page 78
iwcallback.exe	Tells the TeamSite server that the program corresponding to an external or cgi task is finished. Passes a return code to the server.	page 89

iwgetwfobj.exe	Prints the state of any part of the server workflow subsystem in XML.	page 102
iwinvokejob.exe	Starts a job whose instance has already been created on the TeamSite server.	page 109
iwjobc.exe	Creates a job instance (based on a job specification file) on the TeamSite server.	page 110
iwjobvariable.exe	Manipulates workflow variables.	page 111
iwqueryjobs.exe	Prints a list of overall job information based on a query from stdin.	page 125
iwquerytasks.exe	Prints a list of job task information based on a query from stdin.	page 126
iwretrywfop.exe	Retries submission or task update after resolution of conflicts that originally prevented those operations.	page 129
iwrmjob.exe	Unconditionally removes an entire job instance from the TeamSite server.	page 135
iwrmtaskfile.exe	Removes a file from a job task.	page 136
iwrmtaskgroup.exe	Removes groups from group tasks.	page 137
iwrmtaskuser.exe	Removes users from group tasks.	page 138
iwsetjobdescription.exe	Changes the description of a job.	page 140
iwsetjobowner.exe	Changes the owner of a job.	page 141
iwsettaskattrib.exe	Sets a new value for task attributes that have true or false values.	page 142
iwtaketask.exe	Assigns a shared task to a single user.	page 145
iwtaskselect.exe	Marks a usertask finished and selects a successor task.	page 146
iwundochoice.exe	Reverses a user-chosen transition from a user task or group task.	page 147

INTERWOVEN

iwaddtaskfile.exe

Adds a file to a specified task in an instantiated job.

Man Page Group:

Workflow/Job.

Usage:

iwaddtaskfile [-h|-v] [-s *servername*] *taskID path comment*

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
taskID	Specifies the task to which the file will be added. Task IDs are represented as integers.
path	Path relative to area root.
comment	Comment attached to the file.

Example:

The following command adds the file /default/main/WORKAREA/eng/content.txt to
job 7734 on the server production.example.com (assuming the command is issued from
the eng directory):

>iwaddtaskfile -s production.example.com 7734 content.txt

`iwaddtaskgroup.exe`

Adds groups to group tasks.

Man Page Group:

Workflow/Job

Usage:

```
iwaddtaskgroup [-h] [-v] [-s servername] taskID group
```

-h	Print this message.
-v	Print version.
-s *servername*	Use *servername* as TeamSite server.
taskID	ID of task to change.
user	Group to share task.

Example:

>iwaddtaskgroup 343 graphics

INTERWOVEN

`iwaddtaskuser.exe`

Adds users to tasks.

Man Page Group:

Workflow/Job

Usage:

iwaddtaskuser [-h] [-v] [-s *servername*] taskID user

-h	Print this message.
-v	Print version.
-s *servername*	Use *servername* as TeamSite server.
taskID	ID of task to change.
user	User to share task.

Example:
>iwaddtaskuser -s eaglesmith 105 fred

`iwattrib.exe`

Returns metadata information on any object in the TeamSite server, including all branches, workareas, editions, staging areas, files, directories, and symlinks.

Man Page Group:

General Development.

Usage:

To return information on a specific attribute of an object:

```
iwattrib [-h|-v]
iwattrib objectvpath attribute
iwattrib -o objid attribute
```

To list available attributes for an object:

```
iwattrib -l objectvpath
iwattrib -l -o objid
```

-h	Displays usage message.
-v	Displays version.
objectvpath	Vpath of the object (see page 15).
attribute	Attribute to return information about.
lockmodel	Returns the locking model of a branch.
-o *objid*	Objid of the object. An object can be specified by its full vpath or its object ID.
-l	Lists the attributes available for the specified object.

INTERWOVEN

Examples:

>iwattrib -l main

returns the names of all the attributes available for the main branch:

```
cid
objid
crdate
creator
owner
group
name
label
moddate
busy
rootedition
firstedition
lastedition
parentbranch
archive
branchcount
editioncount
workareacount
```

>iwattrib main objid

returns the objid of the main branch:
0x0001025000000000000006d

>iwattrib main archive

returns the objid of the TeamSite archive containing the main branch:
0x0001001000000000000000001

>iwattrib -n main archive

returns the name of the archive containing the main branch:
default

The values of *attribute* available for each type of object are:

Server

Name	Example	Description
cid	0x00000051	Class ID
objid	0x000000000000000000000000a	Object ID
crdate	Thu Aug 24 13:13:30 2000	Creation date
creator	smtp	Uid of the creator
owner	smtp	Uid of the owner
group	root	Gid of the group for sharing
name	IWSERVER	Name
hostname	chocolate	Host name of the TeamSite server
ipaddress	0.0.0.0	IP address of the TeamSite server
verstring	iwserver: 4.5.0 Build 1776 Interwoven 200713	Version of TeamSite running on the TeamSite server

Archive

Name	Example	Description
cid	0x00010010	Class ID
objid	0x000000000000000000000000a	Object ID
crdate	Thu Aug 24 13:13:30 2000	Creation date
creator	smtp	Uid of the creator
owner	smtp	Uid of the owner
group	root	Gid of the group for sharing
name	default	Name
label	Default archive	Comment attached at time of creation

INTERWOVEN

Name	Example	Description
moddate	Thu Aug 24 13:13:30 2000	Date modified
busy	TRUE or FALSE	Tells whether the archive is busy or not
mainbranch	0x00002250000000000000006d	Object id of the main branch

Branch

Name	Example	Description
cid	0x00010250	Class ID
objid	0x00002250000000000000006d	Object ID
crdate	Tue Aug 22 18:11:41 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	dev	Name
label	Main branch	Comment attached at time of creation
moddate	Tue Aug 22 18:11:41 2000	Date modified
busy	TRUE or FALSE	Tells whether the branch is busy or not
rootedition	0x0000201000000000000000077 or INITIAL	Object ID of the edition that the branch is based on (-n returns the name of the edition)
firstedition	0x000020100000000000000051e or INITIAL	Object ID of the first edition on the branch (-n returns the name of the edition)
lastedition	0x0000201000000000000000722 or ed_0001	Object ID of the latest edition on the branch (-n returns the name of the edition)

Name	Example	Description
`parentbranch`	`0x00002250000000000000006d` or `main`	Object ID of the parent branch (`-n` returns the name of the branch)
`archive`	`0x0000202000000000000000001` or `default`	Object ID of the archive (`-n` returns the name of the archive)
`branchcount`	`1`	Number of sub-branches present on the branch (does not recurse to sub-sub branches)
`editioncount`	`2`	Number of editions on the branch
`workareacount`	`4`	Number of workareas on the branch

Edition

Name	Example	Description
`cid`	`0x00010020`	Class ID
`objid`	`0x0000201000000000000000722`	Object ID
`crdate`	`Mon Aug 28 13:45:57 2000`	Creation date
`creator`	`andre`	Uid of the creator
`owner`	`andre`	Uid of the owner
`group`	`marketing`	Gid of the group for sharing
`name`	`ed_0001`	Name
`label`	`baseline edition`	Comment attached at time of creation
`busy`	`TRUE or FALSE`	Tells whether the edition is busy or not
`rootdir`	`0x0000072200000000000000520` or `/`	Object ID of the root directory of the edition (`-n` returns the name of the directory)

INTERWOVEN

Name	Example	Description
nextedition	0x0000201000000000000000722 or ed_0002	Object ID of the edition after the specified edition (-n returns the name of the edition)
prevedition	0x0000201000000000000000051e or INITIAL	Object ID of the edition before the specified edition (-n returns the name of the edition)
branch	0x0000225000000000000000514 or dev	Object ID of the edition's branch (-n returns the name of the branch)
sequenceno	1	Sequence number

Workarea

Name	Example	Description
cid	0x00010100	Class ID
objid	0x0000210000000000000000522	Object ID
crdate	Thu Aug 24 13:13:30 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	andre	Name
label	Andre's workarea	Comment attached at time of creation
moddate	Thu Aug 24 13:13:30 2000	Date modified
modified	TRUE or FALSE	Tells whether the object has been modified or not
busy	TRUE or FALSE	Tells whether the workarea is busy or not

Name	Example	Description
rootdir	0x0000052200000000000000520 or /	Object ID of the root directory of the workarea (-n returns the name of the directory)
baseedition	0x0000201000000000000051e or INITIAL	Object ID of the edition that the workarea is based on (-n returns the name of the edition)
branch	0x0000225000000000000000514 or dev	Object ID of the workarea's branch (-n returns the name of the branch)

Submit Event

Name	Example	Description
cid	0x00002258	Class ID
objid	0x0000225800000000000006bb	Object ID of the submit event
workarea	0x0000210000000000000000522	Object ID of the submitting workarea
submitter	andre	Uid of the user who submitted
sdate	Mon Aug 28 13:45:43 2000	Date of submission
entries	5	Number of files and directories submitted
submit_cmt	first submit to staging area	Submit-level comment
submit_info	keyword	Info field

Update Event

Name	Example	Description
cid	0x0000225a	Class ID
objid	0x0000225a000000000000a9a	Object ID of the update event
workarea	0x0000210000000000000000522	Object ID of the updated workarea

INTERWOVEN

Name	Example	Description
updater	andre	Uid of the user who updated the workarea
udate	Mon Oct 2 16:57:12 2000	Date the workarea was updated
entries	59	Number of files and directories updated

Staging Area

Name	Example	Description
cid	0x00010255	Class ID
objid	0x0000225500000000000000521	Object ID
crdate	Thu Aug 24 13:13:06 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	STAGING	Name
label		Comment attached at time of creation
moddate	Thu Aug 24 13:13:06 2000	Date modified
modified	TRUE or FALSE	Tells whether the object has been modified or not
busy	TRUE or FALSE	Tells whether the staging area is busy or not
rootdir	0x0000052100000000000000520 or /	Object ID of the root directory of the staging area (-n returns the name of the directory)
branch	0x0000225000000000000000514 or dev	Object ID of the staging area's branch (-n returns the name of the branch)

File or Directory

Name	Example	Description
cid	0x08001000	Class ID
objid	0x000005220000058e0000080e	Object ID
crdate	Fri Sep 1 15:55:51 2000	Creation date
creator	andre	Uid of the creator
owner	andre	Uid of the owner
group	marketing	Gid of the group for sharing
name	index.html	Name of the file or directory
label	first submit	Latest comment attached
moddate	Fri Sep 1 16:24:13 2000	Date modified
modified	TRUE or FALSE	Tells whether the object has been modified or not
currdirectory	0x000005220000052000000058e	Object ID of the directory containing the specified item (-n returns the name of the directory)
area	0x0000210000000000000000522	Object ID of the area containing the specified item (-n returns the name of the area)
areacid	0x00002100	Class ID of the area containing the specified item
branch	0x0000225000000000000000514	Object ID of the branch containing the specified item (-n returns the name of the branch)
archive	0x0000202000000000000000001	Object ID of the archive containing the specified item (-n returns the name of the archive)
size	1949	Size of the file or directory (in bytes)
lastuser	andre	User who last submitted the file

INTERWOVEN

Name	Example	Description
istagged	TRUE or FALSE	File is locked or assigned
nfsmode	0x000001ff	NFS permission bits
isdir	TRUE or FALSE	TRUE if the specified item is a directory, FALSE if it is not
taglabel	Locked by launching edit	Comment attached to tag (for example, a lock or assignment)
revisionnumber	3	Ordinal representing the version of the file submitted on a branch
revisionbranch	0x0000225000000000000000514	Object ID of the branch on which this version of the file has been submitted
submitevent	0x0000225800000000000000ab8	Object ID of the submit event for the most recent submission of this file.
history	first submit	Comments associated with file at time of submission (includes both submission comment and individual file comment)
prevversion	0x0000052100000aea00000815	Object ID of the previous version

iwcallback.exe

Notifies the server that the program corresponding to an `<externaltask/>` or `<cgitask/>` is finished and passes the server a return code.

Man Page Group:

Workflow/Job.

Usage:

```
iwcallback [-h|-v] [-s servername] taskID returnCode [comment]
```

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
taskID	Specifies the task to which the file will be added. Task IDs are represented as integers.
returnCode	Specifies the notification given to the server upon program completion. Represented as an integer from 0 to n.
comment	An optional comment.

Example:

The following command sends the return code 2 to the server `production.example.com` when the program named in task 7734 finishes running:

```
>iwcallback -s production.example.com 7734 2
```

INTERWOVEN

iwcat.exe

iwcat is `cat` for TeamSite files. If you specify a vpath to the file, you will see the current version of the file in the area you specify. You can view any previous version of a file by specifying the objid for that version.

Man Page Group:

Version Management.

Usage:

iwcat [-h|-v] *filevpath*

iwcat -o *fileobjid directoryvpath*

-h	Displays usage message.
-v	Displays version.
filevpath	Vpath of the file to display (used to display the current version in an area) (see page 15).
-o *fileobjid*	Objid of the file to display (used to display any version).

Examples:

>iwcat main/WORKAREA/andre/htdocs/about2.html

displays the version of about2.html in workarea andre.

>iwcat -o 0x0000007b0000007d000000bb main/STAGING

displays the version of the file that has the objid 0x0000007b0000007d000000bb (in this case, the second most recent version of about2.html). *directoryvpath* can point to any area containing the file.

`iwckrole.exe`

Checks whether the specified user has the specified role.

Man Page Group:

General Development.

Usage:

```
iwckrole [-h|-v] role user
```

`-h`	Displays usage message.
`-v`	Displays version.
`role`	`author`, `editor`, `admin` or `master`.
`user`	Username of the person whose role you are checking.

Exits with 0 on successful authorization, non-zero on failure.

Example:

>iwckrole admin andre

returns:

YES

INTERWOVEN

`iwcmp.exe`

Compares any two TeamSite areas and reports an exhaustive list of all modifications. For a full description of TeamSite comparison, see the *TeamSite User's Guide*.

Man Page Group:

Version Management.

Usage:

iwcmp [-h|-v] *area1vpath area2vpath*

-h	Displays usage message.
-v	Displays version.
area1vpath	Vpath of area1 (see page 15).
area2vpath	Vpath of area2 (see page 15).

The output is in the form:

status-in-area1 status-in-area2 path-to-file-or-directory

where status is indicated by a combination of the following symbols:

-	File or directory does not exist in this area.
f	File exists.
d	Directory exists.
s	Symlink exists.
+	Item has been modified.
x	File or directory has been deleted.
p	File or directory is marked private.

Example:
>iwcmp main/branch1/WORKAREA/wa1 main/branch1/WORKAREA/wa2

returns:

```
x+      f       htdocs/about.html
f+      -       htdocs/about2.html
d+      -       htdocs/support/
f+      -       htdocs/main.html
f       fp+     htdocs/products.html
```

showing that the file about.html has been renamed to about2.html in the first area, so about.html appears as deleted and about2.html appears as new in the first workarea. A directory named support has been created in the first area and does not exist in the second. The file main.html only exists in the first area specified and has never existed in the second area, and the file products.html exists in both areas, but in the second area it has been marked private and modified.

INTERWOVEN

iwdecode.exe

Decodes HTML-encoded %xx lines to ASCII lines. Receives HTML-encoded lines through stdin.

Man Page Group:

General Development.

Usage:

iwdecode [-h|-v]

        -h   Displays usage message.

        -v   Displays version.

Example:

>iwdecode

Locked%20by%20launching%20edit

^D

returns:

Locked by launching edit

iwdeploy.exe

Deploys website content to the production server. OpenDeploy must be installed for this CLT to be available. For more information on deployment, see the *OpenDeploy Administration Guide*.

Man Page Group:

Deployment.

Usage:

```
iwdeploy [-h] [(-f|-fs) srcConfigFile] [-fd destConfigFile] [-r] [-v]
[-d] [-o package_name] [-t tempFilePath] deployment_name [param=value]+
```

-h	Displays usage message.
(-f|-fs) *srcConfigFile*	Specifies source deployment configuration file (defaults to /etc/iw.deploy.cfg). For information on deployment configuration files, consult Interwoven Professional Services.
	The -f option is provided for backwards-compatibility.
-fd *destConfigFile*	Specifies destination configuration file (defaults to /etc/iw.deployee.cfg). For information about deployment configuration files, consult Interwoven Client Services.
-r	Reverse deployment.
-v	Verbose logging.
-d	Debugging output.
-o *package_name*	Creates a deployment package that can be transferred to the production server by alternate means (for example, via email, manually). This option can only be used with TeamSite-based configurations.
-t *tempFilePath*	Specifies directory path for temporary file created during deployment (needs space for up to ~5 Mb).
deployment_name	Name of the section of *srcConfigFile* to invoke.
[*param=value*]+	Parameters that override configuration file parameters.

INTERWOVEN

`iwencode.exe`

Encodes ASCII lines to HTML-encoded %xx lines. Takes input from `stdin.`

Man Page Group:

General Development.

Usage:

```
iwencode [-h|-v]
```

-h	Displays usage message.
-v	Displays version.

Example:

>iwencode

Locked by launching edit

^D

returns:

```
Locked%20by%20launching%20edit
```

`iwevents.exe`

Returns a history of submissions or updates of a workarea.

Man Page Group:

Workarea Operation.

Usage:

```
iwevents [-h|-v] -s[ubmit]|-u[pdate] [-n[oheader]] [-i[ds]] vpath
latest|all
```

```
iwevents -s[ubmit]|-u[pdate] [-n[oheader]] id
```

`-h`	Displays usage message.
`-v`	Displays version.
`-s`	Gets a history of **Submit** operations.
`-u`	Gets a history of **Get Latest** operations.
`-i`	Displays only the objids of the **Submit** or **Get Latest** events.
vpath	Specifies a vpath to a workarea, edition, or staging area (see page 15).
`latest`	Gets only the latest submit/update.
`all`	Gets all currently logged submits/updates.
id	Specifies the objid for the event.

When `-s` is specified, each entry will have ten fields:

`TIME`	time of submission
`UPDATE`	type of update: `ADD`, `MODIFY`, `DELETE`, `FORCEDELETE`, `FORCEMODIFY`
`TYPE`	`DIRECTORY` or `FILE`
`PATH`	path of the file or directory submitted
`USER`	user who submitted the file or directory
`COMMENT`	comment attached to the file at time of submission
`APPROVE`	`TRUE` or `FALSE`
`EVENTID`	objid of the submit event
`FILEID`	objid of the file
`WORKAREAID`	objid of the workarea

When `-u` is specified, each entry will have eight fields:

TIME	time of update
UPDATE	type of update: ADD, MODIFY, DELETE
TYPE	DIRECTORY or FILE
PATH	path of the file or directory updated
USER	user who performed the update
EVENTID	objid of the update event
FILEID	objid of the file
SOURCEAREAID	objid of the area files were updated from

Examples:

```
>iwevents -s main/WORKAREA/andre all
```

returns a history of all the submissions from workarea "andre" and includes the headers for each entry:

```
TIME                      UPDATE   TYPE          PATH
USER       COMMENT     APPROVE     EVENTID
FILEID                  WORKAREAID

[Fri Apr 21 11:49:13 2000]ADD      DIRECTORY   /htdocs/
andre     <none>      FALSE       0x00010258000000000000000e2
0x0000007a00000079000000e3    0x000101000000000000000007b

[Fri Apr 21 11:59:37 2000]ADD      FILE        /htdocs/CorpGuide2.pdf
andre     <none>      FALSE       0x0001025800000000000011f
0x0000007b000000e400000117    0x000101000000000000000007b

[Fri Apr 21 11:59:37 2000]ADD      FILE        /htdocs/logo.PSD
andre     <none>      FALSE       0x0001025800000000000011f
0x0000007b000000e400000119    0x000101000000000000000007b
```

>`iwevents -s -n main/WORKAREA/andre latest`

returns only the files submitted in the most recent **Submit** operation and strips the headers:

```
[Fri Apr 21 11:59:37 2000]ADD      FILE           /htdocs/CorpGuide2.pdf
andre     <none>      FALSE      0x00010258000000000000011f
0x0000007b000000e400000117 0x00010100000000000000007b

[Fri Apr 21 11:59:37 2000]ADD      FILE           /htdocs/logo.PSD
andre     <none>      FALSE      0x00010258000000000000011f
0x0000007b000000e400000119    0x00010100000000000000007b
```

>`iwevents -u main/WORKAREA/andre all`

returns a list of all files copied in to the workarea during all **Get Latest** operations and includes the headers:

```
TIME                         UPDATE  TYPE        PATH
USER      EVENTID                    FILEID
SOURCEAREAID

[Fri Apr 21 13:34:30 2000]   MODIFYFILE/htdocs/index.html  andre
0x0001025a000000000000013b   0x000001300000007d00000136
0x00010256000000000000007a
```

INTERWOVEN

iwextattr.exe

Enables you to set and look up extended attributes on a file. The file is specified with a full
directory path, including the server. Extended attributes can use any naming scheme, but one
common method is to name attributes by general class, then by their specific description. For
example:

```
description/language=English
description/keywords=marketing, software, enterprise
```

The attributes mentioned above both fall into the descriptive category, but one stores language
information and the other stores keywords. You might also have attributes that describe the file's
relationships to other files, such as its document of origin or the files it links to, for example:

```
/relationships/source
/relationships/links/outbound
/description/keywords
/description/language
```

Man Page Group:

General Development.

Usage:

```
iwextattr [-h|-v] [-s attribute=value|-g attribute|-d attribute|-l]
[-f] vpath
```

Options:

-h	Displays usage message.
-v	Displays version.
-s *attribute=value*	Sets *attribute* to *value*.
-g *attribute*	Gets the value of *attribute*.
-d *attribute*	Deletes *attribute*.
-f	Reads/writes *attribute* value from stdin/stout for -s and -g options.
-l	Lists attributes and values.
vpath	Full path to a file. Uses the form: [//server]/archive{[/branch]+}/{EDITION,WORKAREA}/dirpath

---

If no option is specified, attributes are listed as with `-l`.

When attribute values are printed, unprintable (non-ASCII) characters are outputted as *.

Examples:

>iwextattr -s /description/language=English
Y:\default\main\dev\WORKAREA\andre\products\index.html

sets the `/description/language` attribute on the file `products\index.html` in workarea `andre`.

>iwextattr -l Y:\default\main\dev\WORKAREA\andre\products\index.html

returns a list of all the attributes on `products\index.html`, e.g.:

```
/relationships/source=products.doc
/relationships/links/outbound=/test.html;/mktg/index.html;/eng/index.html

/description/keywords=marketing,software,enterprise
/description/description=Overview of our suite of examples
/description/launch_date=March 19th, 1999
/description/expire_date=May 21st, 1999
/description/language=English
/description/contributors=Mark, Kevin, Stacy
/description/audience=Marketing
```

>iwextattr -g /description/language
Y:\default\main\dev\WORKAREA\andre\products\index.html

returns the value of the `/description/language` attribute on `products\index.html`, for example:

```
English
```

INTERWOVEN

iwgetwfobj.exe

Displays the state of any part of the server workflow subsystem in XML. It can be called in three ways. First, by passing no arguments, in which case it displays the workflow registry. Second, by passing a single job ID, in which case it displays the state of that job. Third, by a task ID, in which case it displays the state of that particular task. Output format is described below in the "DTDs" section.

Man Page Group:

Workflow/Job.

Usage:

iwgetwfobj [-h|-v] [-r] [-s *servername*] [*jobID*|*taskID*]

-h	Displays usage message.
-v	Displays version.
-r	Prints all objects below the specified object recursively.
-s *servername*	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.
jobID	Specifies the job to report information about. Job IDs are represented as integers.
taskID	Specifies the task to report information about. Task IDs are represented as integers.

Example:

The following command generates information about task 7734:

```
>iwgetwfobj 7734
```

DTDs:

Output from `iwgetwfobj` conforms to the following DTDs:

wfregistry

The `<wfregistry/>` element is the repository of all jobs instantiated in the server. The `<activeworkflows/>` element is a list of all jobs actually running.

```
<!ELEMENT wfregistry (workflows, activeworkflows, tasks,
                        activetasks, workflow*)>
    <!ELEMENT workflows (id*)>
        <!ELEMENT id EMPTY>
            <!ATTLIST id v CDATA #REQUIRED>
    <!ELEMENT activeworkflows (id*)>
    <!ELEMENT tasks (id*)>
    <!ELEMENT activetasks (id*)>
```

workflow

When you run `iwgetwfobj` passing a single job ID, you get information about a particular job in the system. The active attribute indicates whether the job is running. The `<tasks/>` element contains the tasks owned by this job. `<starttasks/>` are those tasks that are active at job invocation. `<activetasks/>` are those tasks currently active.

```
<!ELEMENT workflow (description, tasks, starttasks, activetasks,
        locks, events, immediatetasks,
        variables,(updatetask|submittask|usertask
        |externaltask|cgitask|grouptask|endtask)*)>
  <!ATTLIST workflow active (t|f) #REQUIRED
            name NMTOKEN #REQUIRED
            owner CDATA #REQUIRED
            activationtime CDATA #REQUIRED
            creator CDATA #REQUIRED>
  <!ELEMENT description (#PCDATA)>
  <!ELEMENT starttasks (id*)>
  <!ELEMENT locks (lock*)>
    <!ELEMENT lock EMPTY>
        <!ATTLIST lock branch CDATA #REQUIRED
                owner CDATA #REQUIRED
                path CDATA #REQUIRED
```

INTERWOVEN

```
                    workarea CDATA #REQUIRED>
 <!ELEMENT events (event*)>
    <!ELEMENT event (arg*)>
       <!ELEMENT arg EMPTY>
          <!ATTLIST arg v CDATA #REQUIRED>
       <!ATTLIST event date CDATA #REQUIRED
                   name CDATA #REQUIRED
                   other CDATA #REQUIRED
                   othername CDATA #REQUIRED
                   role CDATA #REQUIRED
                   task CDATA #REQUIRED
                   taskname NMTOKEN #REQUIRED
                   user CDATA #REQUIRED>
 <!ELEMENT immediatetasks (id*)>
 <!ELEMENT variables (variable*)>
    <!ELEMENT variable EMPTY>
       <!ATTLIST variable key CDATA #REQUIRED
          value CDATA #REQUIRED>
```

tasks

When you run iwgetwfobj passing a single task ID, you get information about that particular
task in the job. The following are DTDs for the different task types. Most of the attributes and
subelements correspond to the equivalent items in the configuration file.

```
<!ELEMENT updatetask (description, areavpath, activation,
          inactivate, resets, predecessors, files,
          comments, activationtime, unactivationtime,
          successorset, srcareavpath)>
 <!ATTLIST updatetask active (t|f) #REQUIRED
             name NMTOKEN #REQUIRED
             overwritemod (t|f) #REQUIRED
             owner CDATA #REQUIRED
             readonly (t|f) 'f'
             lock (t|f) 'f'
             updatedelete (t|f) #REQUIRED
             needsattention (t|f) 'f'
             owningworkflow CDATA #REQUIRED
             tryingtolock (t|f) #REQUIRED
             unactivatible (t|f) #REQUIRED>
 <!ELEMENT areavpath EMPTY>
```

```
    <!ATTLIST areavpath v CDATA #REQUIRED>
<!ELEMENT activation (pred|or|and|not)?>
    <!ELEMENT or ((pred|or|and|not)*)>
    <!ELEMENT and ((pred|or|and|not)*)>
    <!ELEMENT not (pred|or|and|not)>
    <!ELEMENT pred EMPTY>
        <!ATTLIST pred v CDATA #REQUIRED>
<!ELEMENT inactivate (pred*)>
<!ELEMENT resets (reset*)>
    <!ELEMENT reset EMPTY>
        <!ATTLIST reset v CDATA #REQUIRED>
<!ELEMENT predecessors (predecessor*)>
    <!ELEMENT predecessor EMPTY>
        <!ATTLIST predecessor hasactivated (t|f) #REQUIRED>
        <!ATTLIST predecessor id CDATA #REQUIRED>
<!ELEMENT files (file*)>
    <!ELEMENT file (comments)>
        <!ATTLIST file deleted (t|f) #REQUIRED
                  path CDATA #REQUIRED>
<!ELEMENT comments (comment*)>
    <!ELEMENT comment (#PCDATA)>
        <!ATTLIST comment date CDATA #REQUIRED
                  task CDATA #REQUIRED
                  user CDATA #REQUIRED>
<!ELEMENT activationtime EMPTY>
    <!ATTLIST activationtime v CDATA #REQUIRED>
<!ELEMENT unactivationtime EMPTY>
    <!ATTLIST unactivationtime v CDATA #REQUIRED>
<!ELEMENT successorset (succ*)>
    <!ATTLIST successorset description CDATA #REQUIRED>
    <!ELEMENT succ EMPTY>
        <!ATTLIST succ v NMTOKEN #REQUIRED>
<!ELEMENT srcareavpath EMPTY>
    <!ATTLIST srcareavpath v CDATA #REQUIRED>
```

The needsattention attribute indicates that the submit operation has been attempted but failed because of conflicts. It is up to the user interface to resolve conflicts, and then retry the operation.

INTERWOVEN

Most of the following attributes and subelements are common to all tasks. The `active` attribute tells whether the task is active. The `<owningworkflow/>` element identifies the tasks owning the job. The `<predecessors/>` element shows all possible predecessor tasks and whether they have signalled this task.

```
<!ELEMENT submittask (description, areavpath, activation,
          inactivate, resets, predecessors,
          files, comments, activationtime,
          unactivationtime, successorset)>
  <!ATTLIST submittask active (t|f) #REQUIRED
            name NMTOKEN #REQUIRED
            overrideconflicts (t|f) #REQUIRED
            owner CDATA #REQUIRED
            readonly (t|f) 'f'
            lock (t|f) 'f'
            skipconflicts (t|f) #REQUIRED
            skiplocked (t|f) #REQUIRED
            unlock (t|f) #REQUIRED
            needsattention (t|f) 'f'
            owningworkflow CDATA #REQUIRED
            tryingtolock (t|f) #REQUIRED
            unactivatible (t|f) #REQUIRED>

<!ELEMENT externaltask (description, areavpath, activation,
          inactivate, resets, predecessors, files,
          comments, activationtime, unactivationtime,
          successors, transitionmade, command)>
  <!ATTLIST externaltask active (t|f) #REQUIRED
            name NMTOKEN #REQUIRED
            owner CDATA #REQUIRED
            readonly (t|f) 'f'
            lock (t|f) 'f'
            owningworkflow CDATA #REQUIRED
            tryingtolock (t|f) #REQUIRED
            unactivatible (t|f) #REQUIRED
            undoable (t|f) #REQUIRED>
  <!ELEMENT successors (successorset*)>
  <!ELEMENT command EMPTY>
    <!ATTLIST command v CDATA #REQUIRED>
```

```
<!ELEMENT cgitask (description, areavpath, activation, inactivate,
        resets, predecessors, files, comments,
        activationtime, unactivationtime, successors,
        transitionmade, command)>
  <!ATTLIST cgitask active (t|f) #REQUIRED
          name NMTOKEN #REQUIRED
          owner CDATA #REQUIRED
          readonly (t|f) 'f'
          lock (t|f) 'f'
          owningworkflow CDATA #REQUIRED
          tryingtolock (t|f) #REQUIRED
          unactivatible (t|f) #REQUIRED
          undoable (t|f) #REQUIRED>
```

The undoable attribute in the following sections is turned on for those `<usertask/>` and
`<grouptask/>` elements that have had a successor set chosen for them that can be revoked.
Undoable tasks show up in the user interface. Using the CLT iwundochoice a user can take
back the current choice.

```
<!ELEMENT usertask (description, areavpath, activation, inactivate,
          resets, predecessors, files, comments,
          activationtime, unactivationtime, successors,
          transitionmade)>
    <!ATTLIST usertask active (t|f) #REQUIRED
          name NMTOKEN #REQUIRED
          owner CDATA #REQUIRED
          readonly (t|f) 'f'
          lock (t|f) 'f'
          undoable (t|f) #REQUIRED
          owningworkflow CDATA #REQUIRED
          tryingtolock (t|f) #REQUIRED
          unactivatible (t|f) #REQUIRED>
  <!ELEMENT transitionmade EMPTY>
    <!ATTLIST transitionmade v CDATA #REQUIRED>
```

INTERWOVEN

```
<!ELEMENT grouptask (description, areavpath, activation,
          inactivate, resets, predecessors, files,
          comments, activationtime, unactivationtime,
          successors, transitionmade, users)>
  <!ATTLIST grouptask active (t|f) #REQUIRED
          name NMTOKEN #REQUIRED
          owner CDATA #REQUIRED
          readonly (t|f) 'f'
          lock (t|f) 'f'
          undoable (t|f) #REQUIRED
          owningworkflow CDATA #REQUIRED
          tryingtolock (t|f) #REQUIRED
          unactivatible (t|f) #REQUIRED
          committed (t|f) #REQUIRED>
  <!ELEMENT users (user|group)*>
    <!ELEMENT user EMPTY>
       <!ATTLIST user v CDATA #REQUIRED>
    <!ELEMENT group EMPTY>
       <!ATTLIST group v CDATA #REQUIRED>

<!ELEMENT endtask (description, areavpath, activation, inactivate,
        resets, predecessors, files, comments,
        activationtime, unactivationtime)>
 <!ATTLIST endtask active (t|f) #REQUIRED
          name NMTOKEN #REQUIRED
          owner CDATA #REQUIRED
          owningworkflow CDATA #REQUIRED
          tryingtolock (t|f) #REQUIRED
          unactivatible (t|f) #REQUIRED>
```

`iwinvokejob.exe`

Turns on or runs a job that has been instantiated in the server.

Man Page Group:

Workflow/Job.

Usage:

```
iwinvokejob [-h|-v] [-s servername] jobID
```

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.
jobID	Specifies which job to run. Job IDs are represented as integers.

Example:

The following command starts job 7734 on the server `production.example.com`:

```
>iwinvokejob -s production.example.com 7734
```

INTERWOVEN

iwjobc.exe

Instantiates a job into the server based on a specified job specification file and prints (in decimal) the ID of the job.

Man Page Group:

Workflow/Job.

Usage:

```
iwjobc [-h|-v] [-i] [-s servername] workflowfile
```

-h	Displays usage message.
-v	Displays version.
-i	Invoke job after compilation.
-s *servername*	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.
workflowfile	Specifies the area-relative path name of the workflow specification file.

Example:

The following command instantiates the job defined in `config.xml`:

>iwjobc config.xml

`iwjobvariable.exe`

Manipulates workflow variables.

Man Page Group:

Workflow/Job.

Usage:

```
iwjobvariable [-h|-v] [-s servername]
            (-g key|-c key value|-t key value|-d key) workflowid
```

-h	Print help.
-v	Print version.
-s *servername*	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.
-g *key*	Get value of workflow variable *key*. Prints value to standard output.
-c *key value*	Create variable *key* with value *value*. It is an error if *key* already exists.
-t *key value*	Set variable *key* to value *value*. If variable *key* does not exist, create it.
-d *key*	Delete variable *key*.
workflowid	Specifies the identifier of the workflow.

Example:

>iwjobvariable -g foo

gets the value of the *foo* job variable.

INTERWOVEN

`iwlasted.exe`

Returns the name of the last published edition on a branch.

Man Page Group:

Edition Operation.

Usage:

```
iwlasted [-h|-v] branchvpath
```

-h	Displays usage message.
-v	Displays version.
branchvpath	Vpath of a branch (see page 15).

Example:

>iwlasted main

returns:

```
ed_0001
```

iwlist.exe

Returns list of all areas contained by a parent area.

Man Page Group:
General Development.

Usage:

```
iwlist [-h|-v] [-a|-l|-s] [-m [-u user]] vpath
```

-h	Displays usage message.
-v	Displays version.
-a	Lists all the components of a branch.
-l	Long listing.
-s	Short listing.
-m	Displays all files modifiable by *user*.
-u *user*	Specifies the username of a TeamSite user (defaults to the current user).
vpath	Specifies the vpath of the parent area (see page 15).

Examples:

>iwlist main

returns:

```
branch1
branch2
```

Use the -a tag to include all editions and workareas on the branch.

>iwlist -a main

returns:

```
wa1
wa2
INITIAL
ed_001
branch1
branch2
```

INTERWOVEN

>iwlist -a -l main

returns a full list of objids, comments, and base editions for all sub-branches, workareas, and editions on the branch:

```
wa1
andre's workarea
2205
INITIAL
-----------------------------------------------
wa2
chris's workarea
2205
ed_0001
-----------------------------------------------
-----------------------------------------------
INITIAL
Initial empty edition
0

0
-----------------------------------------------
ed_001

2205
INITIAL
1
-----------------------------------------------
-----------------------------------------------
branch1
test branch
2205
ed_002
-----------------------------------------------
branch2
redesign branch
31001
INITIAL
-----------------------------------------------
```

iwlistlocks.exe

Lists the locks and assignments in a workarea or branch. Displays the following information: lock date, file path, workarea, state, creator/assigner, owner/assignee, comments. To see if a file is locked, see the `istagged` attribute of `iwattrib` (page 88).

Man Page Group:

Workarea Operation.

Usage:

```
iwlistlocks [-h|-v] vpath
```

vpath	Vpath to workarea or branch (see page 15).
-h	Displays usage message.
-v	Displays version.

Example:

>iwlistlocks main

returns a list of all the locks (including assignments) in all workareas on the main branch. The information is in columns—date, path to the file relative to the workarea, workarea containing the lock, status, user who locked the file, owner of the lock, comments—as follows:

```
[Tue Aug 22 10:31:32 2000] /htdocs/about2.html   wa1 LOCKED   andre andre
Locked%20by%20launching%20edit
[Tue Aug 22 10:34:15 2000] /htdocs/banner.gif    wa1 ASSIGNED andre pat
add%20support%20section
[Tue Aug 22 10:34:40 2000] /htdocs/index.html    wa1 DONE     andre pat
added%20new%20links
[Tue Aug 22 10:36:29 2000] /htdocs/products.html wa2 LOCKED   chris chris
<none>
```

INTERWOVEN

iwlistmod.exe

Returns a list of all modified files and directories in the specified area.

Man Page Group:

Workarea Operation.

Usage:

```
iwlistmod [-h|-v] vpath
```

vpath	Vpath to a workarea or directory (see page 15).
-h	Displays usage message.
-v	Displays version.

In the output of `iwlistmod`:

x+ indicates a deleted file or directory

f+ indicates a modified or new file

d indicates a new directory

Examples:

>iwlistmod /default/main/WORKAREA/andre

returns:

```
x+      assets/images/iwhead60.gif
x+      assets/images/iwlogo49.gif
f+      html/support/testing_form.html
f+      html/support/index.html
f+      html/support/instructions.html
d       html/newdir/
f+      index.html
```

>iwlistmod /default/main/WORKAREA/andre/html/support

returns:

```
f+      testing_form.html
f+      index.html
f+      instructions.html
```

`iwlock.exe`

Locks any file or directory in any TeamSite workarea. For a full description of TeamSite locking, see the *TeamSite User's Guide*.

Man Page Group:

Workarea Operation

Usage:

```
iwlock [-f|-s] vpath comment [ownerid] [-f|-s]
```

where *vpath* specifies a vpath to a file.

-h	Displays usage message.
-v	Displays version.
comment	Comment attached to the file.
ownerid	Username of the lock owner.
-f	Force update.
-s	Suppress update.

Example:

iwlock main/branch1/WORKAREA/wa1/test.html 'need to use this for a while'

Locks the file `test.html` in the workarea `wa1` on the sub-branch `branch1`. The comment attached is "need to use this for a while". Because no owner is specified, the lock owner is the user who locked it.

Forcing an update will take the version in the staging area to edit, regardless of whether it is the newer version. Suppressing an update will take the version in the workarea, regardless of whether it is newer than the version in the staging area.

INTERWOVEN

iwlockinfo.exe

Provides detailed information on any lock held in TeamSite, such as who holds the lock and in what workarea.

Man Page Group:

Workarea Operation

Usage:

```
iwlockinfo [-h|-v] vpath
```

where *vpath* specifies a vpath to a file.

–h	Displays usage message.
–v	Displays version.

Example:

>iwlockinfo main/branch1/WORKAREA/wa1/test.html

returns:

```
Name: test.html
Area: wa1
Assignor: andre
Assignee: andre
State: Locked
Comments: fixing some links
```

note `test.html` is locked in workarea `wa1` by Andre, with the comment "fixing some links" attached.

iwlockinfo main/branch1/WORKAREA/wa1/htdocs/index.html

returns:

```
Name: index.html
Area: wa1
Assignor: andre
Assignee: pat
State: Done
Comments: fix the marketing link.
fixed!
```

showing `index.html` in workarea `wa1` has been assigned by Andre to Pat, and that Pat has marked it Done. Comments attached to the file are "fix the marketing link. fixed!

`iwmerge.ipl`

Merges files from one area into another.

Man Page Group:

General Development.

Usage:

```
iwmerge [-n] [-p] [-m|-c] [-a area] [files...]
```

-n	Displays changes without updating files.
-p	Writes merges to stdout.
-m	Resolves non-overlapping merges.
-c	Leaves annotated overlapping merge conflicts in file.
-a area	Area to merge from. Defaults to STAGING.
-h	Prints usage message.
-v	Prints the version string.

iwmkbr.exe

Creates a new TeamSite branch. For more information on creating TeamSite branches, see the *TeamSite User's Guide*.

Man Page Group:

Branch Operation.

Usage:

iwmkbr [*flags*] *parent brname comment rooted* [*ownerid* [*groupid*]]

Creates a new branch in a TeamSite archive.

parent	Vpath of the parent branch of the new branch (see page 15).
brname	Name of the new branch.
comment	Description of the new branch.
rooted	Name of edition in the parent branch to serve as a basis for the new branch.
ownerid	Username of the owner of the new branch.
groupid	Group name of the group of Administrators that has Administrator-level access to the new branch.

Flags:

-h	Displays usage message.
-v	Displays version.
-l[ock=]s	Create branch with submit locking [default].
-l[ock=]o	Create branch with optional write locking.
-l[ock=]m	Create branch with mandatory write locking.

Example:

>iwmkbr //IWSERVER/default/main branch1 'test branch' INITIAL andre

creates a sub-branch called branch1 off of the main branch. The comment attached to this branch is "test branch". It is based on the initial edition of the main branch, and it is owned by user andre. There is no group of Administrators for this branch (only andre has Administrator-level privileges for the branch), and the locking model used is the default, submit locking.

`iwmkwa.exe`

Creates a new TeamSite workarea on a branch. For more information on creating TeamSite workareas, see the *TeamSite User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

`iwmkwa [-h|-v]` *`branchvpath waname comment base_ed`* `[`*`ownerid`* `[`*`groupid`*`]]`

`-h`	Displays usage message.
`-v`	Displays version.
`branchvpath`	Vpath of the branch to create the new workarea on (see page 15).
`waname`	Name of the new workarea.
`comment`	Comment attached to the workarea.
`base_ed`	Specifies which edition on the branch to base the workarea on.
`ownerid`	Specifies who owns the workarea.
`groupid`	Specifies the group that has access to the workarea.

Example:

`>iwmkwa main/branch1 wa1 '' INITIAL chrisc`

This creates a workarea named `wa1` on the sub-branch `branch1`. No comment has been attached to this workarea, and it is based on the initial edition of the branch and owned by `chrisc`. This is a private workarea because no group has access to it.

![INTERWOVEN]

`iwnexted.exe`

Returns the name that would be autogenerated by the `iwpublish -a` command for the next edition on a branch.

Man Page Group:

Edition Operation.

Usage:

`iwnexted [-h|-v]` *branchvpath*

`-h`	Displays usage message.
`-v`	Displays version.
branchvpath	Specifies the vpath of the branch (see page 15).

Example:

if the last edition on the main branch is named `ed_0001`,

>iwnexted main

returns:

`ed_0002`

If the name of the last edition ends with a number, `iwnexted` will increment it. If it does not, `iwnexted` will append a number to the name.

iwprv.exe

Allows users to identify whether any file or directory in any workarea is marked private and automatically mark any file as either private or public. For a full description of private and public files and directories, see the *TeamSite User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

```
iwprv [-h|-v] [-c|-s|-g] list-of-vpath
```

-h	Displays usage message.
-v	Displays version.
-c	Clear the private bit.
-g	Get the private bit (returns 0 if the item is not private, 1 if it is) (default). This option can only accept one vpath.
-s	Set the private bit.
list-of-vpath	List of vpaths of files and directories to mark private (see page 15). Entries must be separated by spaces.

Examples:

>iwprv -s main/br1/WORKAREA/wa1/gifs main/br1/WORKAREA/wa1/mktg main /br1/WORKAREA/wa1/index.html main/br1/WORKAREA/wa1/main.html

makes the directories gifs and mktg and the files index.html and main.html private.

>iwprv -g main/br1/WORKAREA/wa1/gifs
returns: 1 (indicating that the item is private)

>iwprv -c main/br1/WORKAREA/wa1/gifs main/br1/WORKAREA/wa1/mktg main /br1/WORKAREA/wa1/index.html main/br1/WORKAREA/wa1/main.html

makes these same files and directories no longer private.

>iwprv -g main/branch1/WORKAREA/wa1/test.html
returns: 0 (indicating that the item is not private)

INTERWOVEN

iwpublish.exe

Publishes the staging area on a branch. For more information on publishing editions, see the *TeamSite User's Guide*.

Man Page Group:

Edition Operation.

Usage:

iwpublish -g [-w|-s] [-f] *vpath area comment* [*ownerid*]

iwpublish [-h|-v] [-w|-s] [-f] *vpath area edition comment* [*ownerid*]

-g	Generates an edition name based on the last edition name.
-w	Overrides conflicts.
-s	Skips conflicts and files locked in another area.
-f	Fails publish if there have been no changes since the last publish.
-h	Displays usage message.
-v	Displays version.
vpath	Vpath of the branch whose area you want to publish. Can be a full vpath (*//server/archive/branch*) or a relative path (as shown in the example below).
area	The workarea or staging area that you want to publish.
edition	Name of the new edition.
comment	Comment attached to the edition.

Example:

>iwpublish main/branch1 STAGING ed_002 'checkpoint'

publishes the staging area on the sub-branch branch1 as edition ed_002 with the comment "checkpoint" attached.

`iwqueryjobs.exe`

Reads a query from `stdin` and prints to `stdout` a list of job IDs that match the query criteria. Query syntax is described below in the DTD section.

Man Page Group:

Workflow / Job.

Usage:

```
iwqueryjobs [-h|-v] [-s servername]
```

`-h`	Displays usage message.
`-v`	Displays version.
`-s servername`	Specifies the server on which to check for matching query criteria. Server names can be specified by name or IP address.

Example:

The following command searches for matching job IDs on the server `production.example.com`:

`>iwqueryjobs -s production.example.com`

DTD:

All queries must be in XML format that matches the following DTD:

```
<!ELEMENT wfquery (and|or|not|ownedby|active)>
    <!ELEMENT and ((and|or|not|ownedby|active)+)>
    <!ELEMENT or ((and|or|not|ownedby|active)+)>
    <!ELEMENT not (and|or|not|ownedby|active)>
    <!ELEMENT ownedby EMPTY>
        <!ATTLIST ownedby v CDATA #REQUIRED>
    <!ELEMENT active EMPTY>
```

The `ownedby` element returns the job owned by the specified user. The `active` element returns active jobs.

INTERWOVEN

iwquerytasks.exe

Reads a query from `stdin` and prints to `stdout` a list of task IDs that match the query criteria. Query syntax is described below in the "DTD" section.

Man Page Group:

Workflow/Job.

Usage:

iwquerytasks [-h|-v] [-s *servername*]

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the server on which to check for matching query criteria. Server names can be specified by name or IP address.

Example:

The following command searches for matching task IDs on the server `production.example.com`:

>iwquerytasks -s production.example.com

DTD:

All queries must be in XML format that matches the following DTD:

```
<!ELEMENT taskquery (and|or|not|ownedby|active|wfactive|wfownedby|
                      workflow|needsattention|type|sharedby|undoableby)>
    <!ELEMENT and ((and|or|not|ownedby|active|wfactive|wfownedby|
                      workflow|needsattention|type|sharedby|undoableby)+)>
    <!ELEMENT or ((and|or|not|ownedby|active|wfactive|wfownedby|
                      workflow|needsattention|type|sharedby|undoableby)+)>
    <!ELEMENT not (and|or|not|ownedby|active|wfactive|wfownedby|
                      workflow|needsattention|type|sharedby|undoableby)>
    <!ELEMENT ownedby EMPTY>
        <!ATTLIST ownedby v CDATA #REQUIRED>
    <!ELEMENT active EMPTY>
```

```
<!ELEMENT wfactive EMPTY>
<!ELEMENT wfownedby EMPTY>
    <!ATTLIST wfownedby v CDATA #REQUIRED>
<!ELEMENT workflow EMPTY>
    <!ATTLIST workflow v CDATA #REQUIRED>
<!ELEMENT needsattention EMPTY>
<!ELEMENT type EMPTY>
    <!ATTLIST type v (usertask, grouptask, externaltask,
                      cgitask, submittask, updatetask, endtask,
                      dummytask, locktask, wftask) #REQUIRED>
<!ELEMENT sharedby EMPTY>
    <!ATTLIST sharedby v CDATA #REQUIRED>
<!ELEMENT undoableby EMPTY>
    <!ATTLIST undoableby v CDATA #REQUIRED>
```

Expression and Element Descriptions:

Expression/Element	Description
active	Returns tasks which are active.
and, or, and not	Standard logical operations.
needsattention	Returns submit or update tasks that have stalled because of conflicts and cgi tasks that are ready to be run.
ownedby	Returns tasks owned by the specified user.
sharedby	Returns group tasks that can be grabbed by the specified user.
type	Returns tasks of the specified type.
undoable	Returns user or group tasks that can have their chosen transitions taken back.
wfactive	Returns tasks that belong to active jobs.
wfownedby	Returns tasks that belong to jobs owned by the specified user.
workflow	Returns tasks that belong to the specified job.

INTERWOVEN

`iwrename.exe`

Renames any file, directory, workarea, edition, or branch (except `main`) in TeamSite.

Man Page Group:

General Development.

Usage:

`iwrename [-h|-v]` *vpath newname*

`-h`	Displays usage message.
`-v`	Displays version.
vpath	Specifies a vpath to a file, directory, area, or branch (see page 15).
newname	Specifies the new name.

Examples:

>iwrename main/branch1 test

renames the sub-branch `branch1` to `test`.

>iwrename main/test/WORKAREA/wa2 andre

renames the workarea `wa1` on `test` to `andre`.

>iwrename main/test/WORKAREA/andre/htdocs/index.html index2.html

renames the file `index.html` to `index2.html` in workarea `andre`.

`iwretryjobop.exe`

Called when conflicts preventing successful completion of submit or update tasks have been resolved.

Man Page Group:

Workflow/Job.

Usage:

```
iwretryjobop [-h|-v|-o] [-s servername] taskID [comment]
```

-h	Displays usage message.
-v	Displays version.
-o	Force submit or update.
-s *servername*	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
taskID	Specifies the task that will be retried. Task IDs are represented as integers.
comment	An optional comment.

Example:

The following command retries task 7734 on the server `production.example.com`:

```
>iwretryjobop -s production.example.com 7734
```

INTERWOVEN

`iwrevert.exe`

Gives command-line users the same **Revert** functionality that is available from the History screen in the TeamSite GUI. That is, a user can now revert to any version of a file via the command line.

Man Page Group:

Version Management.

Usage:

`iwrevert [-h|-v] vpath rev`

`-h`	Displays usage message.
`-v`	Displays version.
`vpath`	Specifies a vpath to a file.
`rev`	Specifies the version of the file to revert to. `rev` can have any of these formats:
	A revision string, as displayed by `iwrlog` or the History screen in the TeamSite GUI (for example, `/main/br/52`)
	A number, indicating how many revisions back (for example, `3` to revert to three versions ago)
	The object id of the version of the file (obtainable through `iwrlog`).

iwrlog.exe

Shows a revision log for the specified TeamSite file. Provides functionality similar to RCS's `rlog` for TeamSite files.

Man Page Group:

Version Management.

Usage:

`iwrlog [-h|-v][-c][-m] `*`vpath`*

`-h`	Displays usage message.
`-v`	Displays version.
`-c`	Omits comments/labels from listing.
`-m`	Displays log information in machine-parsable format. Paths and comments are URL-encoded; fields are separated by tabs.
vpath	Vpath to a file (see page 18).

Example:

>iwrlog -c /default/main/branch1/WORKAREA/andre/htdocs/index.html

returns the revision number, date, user, and area containing that version for each version of the file:

```
file /default/main/branch1/WORKAREA/andre/htdocs/index.html (locked)
working revision /main/branch1/6+
----------------------------
name /htdocs/index.html
revision /main/branch1/6+
last modified Wed Jul 26 11:26:13 2000  by andre in area andre
size 16911
objid 0x0000478f0000479200004816
submit event objid N/A
----------------------------
name /htdocs/index.html
revision /main/branch1/6
last modified Wed Jul 26 11:29:19 2000  by andre in area ed_0002
size 16919
objid 0x000048180000479d0000480f
submit event objid 0x0000225800000000000480e
```

INTERWOVEN

```
--------------------------
name /htdocs/index.html
revision /main/branch1/5
last modified Wed Jul 26 11:26:13 2000  by andre in area ed_0001
size 16911
objid 0x000048060000479d00004805
submit event objid 0x00002258000000000004802
--------------------------
name index.html
revision /main/branch1/4
last modified Wed Jul 26 11:24:35 2000  by andre
size 16904
objid 0x00004806000048040000047fa
submit event objid 0x000022580000000000047f7
--------------------------
name index.html
revision /main/branch1/3
last modified Tue Jul 25 18:59:51 2000  by andre
size 1206
objid 0x00004806000047f9000047b7
submit event objid 0x000022580000000000047b4
--------------------------
name index.html
revision /main/branch1/2
last modified Tue Jul 25 18:58:46 2000  by andre
size 4
objid 0x00004806000047b6000047ac
submit event objid 0x000022580000000000047a9
--------------------------
name index.html
revision /main/branch1/1
last modified Tue Jul 25 18:39:17 2000  by andre
size 0
objid 0x00004806000047ab000047a1
submit event objid 0x00002258000000000000479b
==================================================================
total revisions: 7
==================================================================
```

`iwrmbr.exe`

Removes the specified branch and all of its contents from TeamSite. For more information on deleting TeamSite branches, see the *TeamSite User's Guide*.

Man Page Group:

Branch Operation.

Usage:

```
iwrmbr [-h|-v] parentvpath brname
```

```
iwrmbr [-h|-v] branchvpath
```

branchvpath	Vpath of the branch to be deleted (see page 15).
parentvpath	Vpath of the parent branch (see page 15).
brname	Name of the branch to be deleted.
-h	Displays usage message.
-v	Displays version.

Example:

>iwrmbr main branch1

removes the sub-branch `branch1` from the main branch.

INTERWOVEN

`iwrmed.exe`

Deletes any edition on a branch. For more information on deleting TeamSite editions, see the *TeamSite User's Guide*.

Man Page Group:

Edition Operation.

Usage:

`iwrmed [-h|-v]` *`branchvpath edition`*

`-h`	Displays usage message.
`-v`	Displays version.
`branchvpath`	Vpath of the branch on which the edition is located (see page 15).
`edition`	Name of the edition to be removed.

Example:

`>iwrmed main/branch1 ed_001`

removes the edition `ed_001` from the sub-branch `branch1`.

`iwrmjob.exe`

Unconditionally removes a job instance from the server. Only use as a last resort.

Man Page Group:

Workflow/Job.

Usage:

```
iwrmjob [-h|-v] [-s servername] jobID
```

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
jobID	Specifies the job that will be removed. Job IDs are represented as integers.Multiple job IDs may be specified.

Example:

The following command removes job 7734 from the server `production.example.com`:

```
>iwrmjob -s production.example.com 7734
```

INTERWOVEN

`iwrmtaskfile.exe`

Removes a file from a specific task in an instantiated job.

Man Page Group:

Workflow/Job.

Usage:

iwrmtaskfile [-h|-v] [-s *servername*] *taskID file*

-h	Displays usage message.
-v	*Displays version.*
-s *servername*	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
taskID	Specifies the task from which the file will be removed. Task IDs are represented as integers.
file	Specifies the area-relative path of the file that will be removed from the job task.

Example:

The following command removes the file /default/main/WORKAREA/eng/content.txt from job 7734 on the server production.example.com (assuming the command is issued from the eng directory):

>iwrmtaskfile -s production.example.com 7734 content.txt

`iwrmtaskgroup.exe`

Removes groups from group tasks.

Man Page Group:

Workflow/Job

Usage:

```
iwrmtaskgroup [-h] [-v] [-s servername] taskID group
```

`-h`	Print this message.
`-v`	Print version.
`-s servername`	Use *servername* as TeamSite server.
`taskID`	ID of task to change.
`user`	Group to remove from task.

Example:
>iwrmtaskgroup -s darkstar 1973 approvers

INTERWOVEN

`iwrmtaskuser.exe`

Removes users from group tasks.

Man Page Group:

Workflow/Job

Usage:

`iwrmtaskuser [-h] [-v] [-s servername] taskID user`

`-h`	Print this message.
`-v`	Print version.
`-s servername`	Use *servername* as TeamSite server.
`taskID`	ID of task to change.
`user`	User to remove from task.

Example:

`>iwrmtaskuser -s factotum 420 bukowski`

`iwrmwa.exe`

Removes the specified workarea. For more information on deleting workareas, see the *TeamSite User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

`iwrmwa [-h|-v]` *branchvpath workarea*

`-h`	Displays usage message.
`-v`	Displays version.
branchvpath	Vpath of the branch on which the workarea is located (see page 15).
workarea	Name of the workarea to be removed.

Example:

>iwrmwa main/branch1 wa1

removes the workarea `wa1` from the sub-branch `branch1`.

INTERWOVEN

`iwsetjobdescription.exe`

Changes the description of a job.

Man Page Group:

Workflow/Job

Usage:

`iwsetjobdescription [-h] [-v] [-s *servername*] workflowID description`

`-h`	Print this message.
`-v`	Print version.
`-s *servername*`	Use *servername* as TeamSite server.
`workflowID`	ID of job to change.
`description`	New description for job.

Example:

>iwsetjobdescription 344 2002_press_releases

`iwsetjobowner.exe`

Changes the owner of a job.

Man Page Group:

Workflow/Job

Usage:

```
iwsetjobowner [-h] [-v] [-s servername] workflowID owner
```

-h	Print this message.
-v	Print version.
-s *servername*	Use *servername* as TeamSite server.
workflowID	ID of job to change.
owner	New owner for job.

Example:

>iwsetjobowner -s bevis 112 nickSaloman

INTERWOVEN

iwsettaskattrib.exe

Sets a new value for task attributes that have true or false values (for example, readonly or lock).

Man Page Group:

Workflow/Job

Usage:

iwsettaskattrib [-h] [-v] [-s *servername*] taskid name value

-h	Print this message.
-v	Print version.
-s *servername*	Use *servername* as TeamSite server.
taskid	ID of task to change.
name	Name of attribute to modify.
value	New value for attribute.

Example:
>iwsettaskattrib -s maynard 462 readonly false

iwsubmit.exe

Submits any TeamSite workarea, file, or directory to the staging area. For a full description of submitting files to the staging area, see the *TeamSite User's Guide*.

Man Page Group:

Workarea Operation.

Usage:

```
iwsubmit [-h] [-v] [-s] [-w|-x] [-u] [-r] [-c global_comment]
[-i info_comment] {vpath file_comment}+

iwsubmit [-h] [-v] [-s] [-w|-x] [-u] [-r] [-c global_comment]
[-i info_comment] -f file_name
```

-c *global_comment*	Comment to attach to the submit event.
-i *info_comment*	Second comment to attach to the submit event.
vpath	One or more vpaths to a file, directory, or workarea (see page 15).
file_comment	Comment to attach to the file, directory, or workarea specified.
-f *file_name*	File from which to read the vpath and *file_comment*. Use newline to separate vpath and file_comment). If *file_name* is -, the information is read from stdin.

Flags:

-h	Displays usage message.
-v	Displays version.
-s	Skips conflicts and locked files.
-w	Overwrites all files. Files in STAGING but not in *src_area* are deleted.
-x	Overwrites conflicts.
-u	Unlocks locked files.
-r	Reports on submitted files.

INTERWOVEN

Example:

```
>iwsubmit -w -u -c 'updated support info' -i 'keyword' main/WORKAREA/
andre/htdocs/index.html 'added imagemap'
```

submits the file `index.html` to the staging area, with the comment "added imagemap" attached to the file. The comments for the submit operation are "updated support info" and "keyword". If `index.html` is locked, this operation will unlock it. If the version in the staging area is newer, this operation will overwrite it.

`iwtaketask.exe`

Assigns a shared task to a specified user.

Man Page Group:

Workflow/Job.

Usage:

```
iwtaketask [-h|-v] [-s servername] taskID username
```

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
taskID	Specifies the task that will be assigned. Task IDs are represented as integers.
username	Specifies the user who will receive the task assignment.

Example:

The following command assigns task 7734 on the server `production.example.com` to user Andre:

>iwtaketask -s production.example.com 7734 Andre

INTERWOVEN

iwtaskselect.exe

Marks a usertask finished and chooses which `<successorset/>` element to signal.

Man Page Group:

Workflow/Job.

Usage:

```
iwtaskselect [-h|-v] [-s servername] taskID choice [comment]
```

`-h`	Displays usage message.
`-v`	Displays version.
`-s servername`	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
`taskID`	Specifies the task that will be marked finished. Task IDs are represented as integers.
`choice`	Specifies the index (starting from 0) of the `<successorset/>` element in the `<successors/>` section of the task's configuration.
`comment`	Optional comment.

Example:

The following command marks task 7734 done on the server `production.example.com` and signals the third `<successorset/>` element in that task's `<successors/>` section:

>iwtaskselect -s production.example.com 7734 2

`iwundochoice.exe`

When a user chooses a transition from a usertask or grouptask, that choice can be taken back. `iwundochoice` reverses one such task. The `iwundochoice` CLT cannot reverse a transition for a task with multiple successors.

Man Page Group:

Workflow/Job.

Usage:

`iwundochoice [-h|-v] [-s servername] taskID`

`-h`	Displays usage message.
`-v`	Displays version.
`-s servername`	Specifies the server on which the job is instantiated. Server names can be specified by name or IP address.
`taskID`	Specifies the task that was active prior to the transition. Task IDs are represented as integers.

Example:

The following command reverses the transition from task 7734 on the server `production.example.com`:

`>iwundochoice -s production.example.com 7734`

INTERWOVEN

iwunlock.exe

Unlocks any file or directory in any TeamSite workarea. See the *TeamSite User's Guide* for a full description of unlocking files.

Man Page Group:

Workarea Operation.

Usage:

iwunlock [-h|-v] *vpath*

-h	Displays usage message.
-v	Displays version.
vpath	Specifies a vpath to a file.

Example:

iwunlock main/branch1/WORKAREA/wa1/index.html

Unlocks the file index.html in the workarea wa1 on the sub-branch branch1. Only one file may be unlocked at a time with this command.

iwupdate.exe

Updates any TeamSite file, directory, or workarea with the latest version of any file system element. This command acts like the **Copy To** or (if *vpath_from* specifies the staging area on that branch) the **Get Latest** command in the GUI. See the *TeamSite User's Guide* for more information about these commands.

Man Page Group:

Workarea Operation.

Usage:

```
iwupdate [-w|-s|-x] [-o] [-r] {src_vpath}+ dst_vpath
iwupdate [-w|-s|-x] [-o] [-r] [-f vpath_file] dst_vpath
```

Where *src_vpath* specifies a path to a file, directory, or area and *dst_vpath* specifies a path to a workarea (*dst_vpath* may specify a path to a file or directory in a workarea if the area-relative portions of *dst_vpath* and *src_vpath* are the same).

-h	Displays usage message.
-v	Displays version.
-w	Overwrites all files. Files in *dst_area* but not in *src_area* are deleted.
-x	Overwrites all conflicts.
-s	Overwrites all versioned files.
-r	Reports on updated files.
-f	Reads source vpaths from the file *vpath_file* (if *vpath_file* is -, STDIN will be read).
-o	Directory update conflicts return success.
src_vpath	Specifies a path to a file, directory, or area.
dst_vpath	Specifies a path to a workarea (*dst_vpath* may specify a path to a file or directory in a workarea if the area relative portions of *dst_vpath* and *src_vpath* are the same).

Example:

>iwupdate -x main/branch1/WORKAREA/wa1 main/branch1/WORKAREA/wa2

copies all the files that are different in workarea wa1 to workarea wa2, overriding any conflicts.

INTERWOVEN

iwvpath.exe

Prints all or parts of the version path of the specified object.

Man Page Group:

General Development.

Usage:

```
iwvpath [-h|-v] [-p|-b|-d|[-s][-a]] objectvpath
```

```
iwvpath [-p|-b|-d|[-s][-a]] -o objid
```

`objectvpath`	Vpath of any TeamSite object (see page 15).
`-o objid`	Objid of any TeamSite object (see page 19).
`-a`	Prints the area vpath (rooted at the archive) (see page 15).
`-b`	Prints the branch relative vpath only (see page 15).
`-d`	Prints the directory path only (see page 15).
`-p`	Prints the components of the version path in separate lines:

```
<server>
<archive>
<branch>
<area-type>
<area-name>
<directory-path>
```

where `<area-type>` is WORKAREA, STAGING, or EDITION

`-h`	Displays usage message.
`-v`	Displays version.
`-s`	Includes the server as part of the vpath.

Examples:

>iwvpath -d main/WORKAREA/andre/htdocs/index.html

returns:

```
/htdocs/index.html
```

>iwvpath -p -c fse -o 0x0000007b0000007d00000093 //IWSERVER/default

returns:

```
chocolate
default
/main
WORKAREA
andre
/htdocs/index.html
```

>iwvpath -b -c fse -o 0x0000007b0000007d00000093 //IWSERVER/default

returns:

```
/main
```

>iwvpath -d -c fse -o 0x0000007b0000007d00000093 //IWSERVER/default

returns:

```
/htdocs/index.html
```

Chapter 4

Command Triggers

This chapter describes how to:

• Configure command triggers to run as services.

• Start and stop command triggers that have been set up to run as services.

This chapter also contains man pages for all supported command triggers. All man pages in this chapter are presented as one group, arranged alphabetically:

Trigger	Description	See...
`iwat.ipl`	Registers an event handler for events that do not have a specific command trigger.	page 159
`iwatcreate.ipl`	Triggers a custom script each time a file or a directory is created.	page 160
`iwatlock.ipl`	Triggers a custom script each time a file or a directory is locked.	page 161
`iwatmkbr.ipl`	Triggers a custom script each time a new branch is created.	page 162
`iwatmkwa.ipl`	Triggers a custom script each time a new workarea is created.	page 163
`iwatpub.ipl`	Triggers a custom script each time a new edition is published.	page 164
`iwatrmbr.ipl`	Triggers a custom script each time a branch is deleted.	page 165
`iwatrmed.ipl`	Triggers a custom script each time an edition is deleted.	page 166
`iwatrmwa.ipl`	Triggers a custom script each time a workarea is deleted.	page 167
`iwatserver.ipl`	Triggers a custom script each time a StartUp, ShutDown, Freeze, Thaw, or DiskLow event occurs.	page 168
`iwatsub.ipl`	Triggers a custom script each time a file or a directory is submitted.	page 169
`iwatunlock.ipl`	Triggers a custom script each time a file or a directory is unlocked.	page 170

INTERWOVEN

iwatupdate.ipl	Triggers a custom script each time a Copy To or Get Latest update event occurs.	page 171
iwlsat.ipl	Lists registered `iwat` programs.	page 172
iwrmat.ipl	Removes registered `iwat` programs.	page 173

Starting Command Triggers

TeamSite command triggers run custom scripts whenever certain events occur in the TeamSite system. For example, the `iwatpub` trigger can be configured to execute an email notification script when an edition is published. Each trigger that is invoked tails the event log and responds to different events. See Appendix B, "Sample Command Trigger Scripts," for script examples.

Because they are non-terminating, command triggers are usually included in a script that starts and stops the process and its tail.

By default, the `iwat` command triggers invoke user scripts synchronously. That is, they wait for the scripts to return before handling the next event. You can use the syntax shown in the following example to configure an `iwat` trigger to run a script asynchronously:

```
iwatserver.ipl "progname progargs ...  &"
```

Note that the ampersand is inside the quotation marks that encase the command. This prevents the shell from running `iwatserver` itself in the background. As coded above, *progname* runs in the background when executed. The `iwatserver` command trigger is just used as an example here. You can use this syntax with any `iwat` command trigger.

To configure the automatic triggering of custom scripts, you will need to use `srvany.exe` and `instsrv.exe` (available as part of the Windows NT Resource Kit).

Notes:

- In the following steps, you must use MS-DOS naming conventions if the file or directory names that you specify contain spaces or more than eight characters. For example, instead of C:\iw-home\`Program Files\Interwoven`, you would specify C:\iw-home\`Progra~1\Interw~1`. You can use the `dir /x` command to display the long and short versions of the file names in the current directory.

- The `String` data type in `regedit` is the same as REG_SZ in `regedt32`.

- See the Windows NT Resource Kit document `srvany.doc` for more information about `srvany`.

Perform the following procedure to set up a command trigger script as a service:

1. Create a service:

   >instsrv.exe <*servicename*> "C:*iw-home*\tools\srvany.exe"
   For example:
   >instsrv.exe iwatsub "C:*iw-home*\tools\srvany.exe"

2. Start `regedt32.exe` (note that `regedt32` does not contain the letter `i`). Go to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ *servicename*. Using the example from Item 1, *servicename* would be **iwatsub**.

3. Add a key called `Parameters`.

4. Add an `Application` value to the `Parameters` key:

   `Application: REG_SZ:` *script*

   where *script* specifies the full path to your executable, including extension. Continuing with the example from Item 1, this line would read as follows for a perl script named `iwperl.exe` residing in the `iw-perl/bin` directory:

   `Application: REG_SZ: C:\`*iw-home*`\iw-perl\bin\iwperl.exe`

5. Add an `AppParameters` value to the `Parameters` key:

   `AppParameters: REG_SZ: C:\`*iw-home*`\bin\`*triggername*`.ipl` *script* [*parameters*]
   where *triggername* is the command trigger you want to run when the submit event occurs and *parameters* are the parameters (if any) the script requires that can be known at service startup time. For example (assuming no parameters are necessary):

INTERWOVEN

```
AppParameters: REG_SZ: C:\iw-home\bin\iwatsub.ipl C:\iw-home\
iw-perl\bin\iwperl.exe
```

6. Add an `AppDirectory` value to the `Parameters` key:

    ```
    AppDirectory: REG_SZ: exec_dir
    ```
    where `exec_dir` is the directory where you want the application to execute. For example:
    ```
    C:\iw-home\iw-perl\bin
    ```

To remove a service:

> **>instsrv.exe *servicename* REMOVE**

To start a service:

> **>net start *servicename***

To stop the service:

> **>net stop *servicename***

Environment Variables

When using command triggers, environment variables are set before the program *progname* is run. The events and their values are described under the appropriate command trigger.

The IW_EVENT environment variable has one of the following values, depending on the command trigger being used. All events, including those without an entry in the Command Trigger column, can be monitored using iwat.

IW_EVENT Environment Variable	Description	Command Trigger
CreateBranch	A branch has been created	iwatmkbr
CreateFSE	An FSE (file/directory/symlink) has been created	iwatcreate
CreateWorkarea	A workarea has been created	iwatmkwa
DeleteEA	An extended attribute (EA) has been deleted from a file	
DestroyBranch	A branch has been deleted	iwatrmbr
DestroyEdition	An edition has been deleted	iwatrmed
DestroyFSE	An FSE has been deleted	
DestroyWorkarea	A workarea has been deleted	iwatrmwa
DiskFail	An I/O error has occurred on the disk(s) containing the backing store.	iwatserver
DiskLow	The disk(s) containing the backing store are running low on space.	iwatserver
Freeze	The server has been frozen (iwfreeze)	iwatserver
Lock	A file or directory has been locked	iwatlock
MarkDoneFile	A file has been marked as "done"	
ModifyFSE	An FSE has been modified	
PublishStagingArea	A new edition has been published from the staging area	iwatpub
ReassignFile	A file has been reassigned	

INTERWOVEN

IW_EVENT Environment Variable	Description	Command Trigger
RejectFile	A file has been reject	
RenameFSE	An FSE has been renamed	
ResetConfig	TeamSite has been asked to reload its configuration files (`iwreset`)	
SetEA	An EA has been added to/modified on a file	
ShutDown	The server has been shut down	`iwatserver`
Startup	The server has been started	`iwatserver`
Submit	A file or directory has been submitted	`iwatsub`
SyncDestroy	A file with EAs has been deleted	
SyncModify	A file with EAs has been modified	
SyncRevert	A file with EAs has reverted to an earlier version	
Thaw	The server has been unfrozen (thawed)	`iwatserver`
UnassignFile	A file has been unassigned	
Unlock	A file or directory has been unlocked	`iwatunlock`
Update	A file or directory has been updated	`iwatupdate`

`iwat.ipl`

This program is used to register an event handler for events that do not have a specific command trigger program.

Usage:

```
iwat.ipl [-h|-v] [-s servername] eventregex_program
```

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.
eventregex_program	A regular expression to match complete iwevents.log lines. When an event matches this regular expression, the program is run.

Example:

>iwat CreateWorkarea /usr/ucb/echo

INTERWOVEN

`iwatcreate.ipl`

`iwatcreate` is a non-terminating script which executes the custom script *`progname`* once each time a file or a directory has been created.

Usage:

`iwatcreate.ipl [-h|-v] [-p] progname progargs...`

`-h`	Displays usage message.
`-v`	Displays version.
`-p`	Prints process ID before starting.
`progname`	Custom script to be executed.
`progargs`	Arguments to *`progname`*.

The following environment variables are set (to the described values) before the program *`progname`* is run:

`IW_EVENT`	"CreateFSE"
`IW_TIMESTAMP`	Time when the event occurred.
`IW_EVENTUSER`	Username of person who performed the operation.
`IW_EVENTROLE`	Role `IW_EVENTUSER` was in during the operation.
`IW_FILE`	vpath of file (rooted at the archive).

`iwatlock.ipl`

iwatlock is a non-terminating script which executes the custom script *progname* once each time a file or a directory is locked.

Usage:

```
iwatlock.ipl [-h|-v] [-p] progname progargs...
```

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"Lock"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_FILE	vpath of file (rooted at the archive).
IW_EDITOR	Workarea owner.

INTERWOVEN

iwatmkbr.ipl

iwatmkbr is a non-terminating script which executes the custom script *progname* once each time a new branch is created.

Usage:

iwatmkbr.ipl [-h|-v] [-p] *progname progargs...*

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"CreateBranch"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_BRANCH	vpath of branch (rooted at the archive).

`iwatmkwa.ipl`

iwatmkwa is a non-terminating script which executes the custom script *progname* once each time a new workarea is created.

Usage:

iwatmkwa.ipl [-h|-v] [-p] *progname progargs...*

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"Create Workarea"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_WORKAREA	vpath of workarea (rooted at the archive).

INTERWOVEN

`iwatpub.ipl`

iwatpub is a non-terminating script which executes the custom script *progname* once each time a new edition is published.

Usage:

iwatpub.ipl [-h|-v] [-p] *progname progargs...*

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"PublishStagingArea"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_EDITION	vpath of edition (rooted at the archive).
IW_STAGINGAREA	Version path of the staging area published.

`iwatrmbr.ipl`

`iwatrmbr` is a non-terminating script which executes the custom script *`progname`* once each time a branch is deleted.

Triggers an event (custom script) upon removal of a branch.

Usage:

`iwatrmbr.ipl [-h|-v] [-p]` *`progname progargs...`*

`-h`	Displays usage message.
`-v`	Displays version.
`-p`	Prints process ID before starting.
`progname`	Custom script to be executed.
`progargs`	Arguments to *`progname`*.

The following environment variables are set (to the described values) before the program *`progname`* is run:

`IW_EVENT`	"DestroyBranch"
`IW_TIMESTAMP`	Time when the event occurred.
`IW_EVENTUSER`	Username of person who performed the operation.
`IW_EVENTROLE`	Role `IW_EVENTUSER` was in during the operation.
`IW_BRANCH`	vpath of branch (rooted at the archive).

INTERWOVEN

`iwatrmed.ipl`

`iwatrmed` is a non-terminating script which executes the custom script *`progname`* once each time an edition is deleted.

Usage:

`iwatrmed.ipl [-h|-v] [-p]` *`progname progargs...`*

`-h`	Displays usage message.
`-v`	Displays version.
`-p`	Prints process ID before starting.
`progname`	Custom script to be executed.
`progargs`	Arguments to *`progname`*.

The following environment variables are set (to the described values) before the program *`progname`* is run:

`IW_EVENT`	"DestroyEdition"
`IW_TIMESTAMP`	Time when the event occurred.
`IW_EVENTUSER`	Username of person who performed the operation.
`IW_EVENTROLE`	Role `IW_EVENTUSER` was in during the operation.
`IW_EDITION`	vpath of edition (rooted at the archive).

`iwatrmwa.ipl`

iwatrmwa is a non-terminating script which executes the custom script *progname* once each time a workarea is deleted.

Usage:

iwatrmwa.ipl [-h|-v] [-p] *progname progargs...*

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"DestroyWorkarea"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_WORKAREA	vpath of workarea (rooted at the archive).

INTERWOVEN

`iwatserver.ipl`

The `iwatserver` CLT is a non-terminating script that executes *progname* once each time after one of the following events has occurred: StartUp, ShutDown, Freeze, Thaw, DiskLow, or DiskFail.

Usage:

```
iwatserver.ipl [-h|-p|-v] progname progargs...
```

`-h`	Prints usage message.
`-p`	Prints the process ID before starting.
`-v`	Prints the version string.
`progargs...`	Arguments to *progname*.

The following environment variables are set (to the described values) before *progname* is run:

`IW_EVENT`	One of the events: StartUp, ShutDown, Freeze, Thaw, DiskLow, or DiskFail.
`IW_TIMESTAMP`	The time when the event occurred.
`IW_EVENTUSER`	Username of person who performed the operation.
`IW_EVENTROLE`	Role that `IW_EVENTUSER` was in during the operation.
`IW_COMMENTS`	Comments associated with the event.

`iwatsub.ipl`

iwatsub is a non-terminating script which executes the custom script *progname* once each time a file or a directory is submitted.

Triggers an event (custom script) upon submission of any file system element.

Usage:

iwatsub.ipl [-h|-v] [-p] *progname progargs...*

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"Submit"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTLINE	Copy of the line from the event log that originally triggered the event.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_WORKAREA	vpath of workarea (rooted at the archive) (same as IW_AREA and IW_VPATH).
IW_SUBMITID	objid associated with the submit event.
IW_EDITOR	Workarea owner.

INTERWOVEN

`iwatunlock.ipl`

iwatunlock is a non-terminating script which executes the custom script *progname* once each time a file or a directory is unlocked.

Usage:

iwatunlock.ipl [-h|-v] [-p] *progname progargs...*

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"Unlock"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_FILE	vpath of file (rooted at the archive).
IW_EDITOR	Workarea owner.

`iwatupdate.ipl`

iwatupdate is a non-terminating script which executes the custom script *progname* once each time a Copy To or Get Latest update event occurs.

Triggers a custom script each time.

Usage:

`iwatupdate.ipl [-h|-v] [-p] progname progargs...`

-h	Displays usage message.
-v	Displays version.
-p	Prints process ID before starting.
progname	Custom script to be executed.
progargs	Arguments to *progname*.

The following environment variables are set (to the described values) before the program *progname* is run:

IW_EVENT	"Update"
IW_TIMESTAMP	Time when the event occurred.
IW_EVENTUSER	Username of person who performed the operation.
IW_EVENTROLE	Role IW_EVENTUSER was in during the operation.
IW_WORKAREA	Version path of workarea (rooted at the archive).
IW_UPDATEID	objid associated with the update event.
IW_EDITOR	Workarea owner.

INTERWOVEN

iwlsat.ipl

iwlsat lists registered iwat programs, their IDs, and their event regular expressions.

Usage:

iwlsat.ipl [-h|-v] [-s *servername*]

-h	Displays usage message.
-v	Displays version.
-s *servername*	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.

Example:

>iwlsat

displays the following output:

ID: 1234   "CreateWorkarea"   /usr/ucb/echo

`iwrmat.ipl`

`iwrmat` unregisters programs that have been added using the `iwat` command.

Usage:

`iwrmat.ipl [-h|-v] [-s servername] ID`

`-h`	Displays usage message.
`-v`	Displays version.
`-s servername`	Specifies the TeamSite server to which to connect. It can be specified by name or IP address. The default value is the current TeamSite server.
`ID`	The ID number of the `iwat` program to unregister.

Example:
`>iwrmat 1234`

Appendix A

Master List

The following is an alphabetized master list of all the CLTs and command triggers described in this manual:

CLT/Trigger	Description	See...
`iwabort.exe`	Provides a method for terminating a long-running server operation.	page 24
`iwaddtaskfile.exe`	Adds a file to a job task that is part of a job instance already created on the TeamSite server.	page 76
`iwaddtaskgroup.exe`	Adds groups to group tasks.	page 77
`iwaddtaskuser.exe`	Adds users to tasks.	page 78
`iwadduser.ipl`	Adds a user to TeamSite role files.	page 26
`iwat.ipl`	Registers an event handler for events that do not have a specific command trigger.	page 159
`iwatcreate.ipl`	Triggers a custom script each time a file or a directory is created.	page 160
`iwatlock.ipl`	Triggers a custom script each time a file or a directory is locked.	page 161
`iwatmkbr.ipl`	Triggers a custom script each time a new branch is created.	page 162
`iwatmkwa.ipl`	Triggers a custom script each time a new workarea is created.	page 163
`iwatpub.ipl`	Triggers a custom script each time a new edition is published.	page 164
`iwatrmbr.ipl`	Triggers a custom script each time a branch is deleted.	page 165
`iwatrmed.ipl`	Triggers a custom script each time an edition is deleted.	page 166

INTERWOVEN

`iwatrmwa.ipl`	Triggers a custom script each time a workarea is deleted.	page 167
`iwatserver.ipl`	Triggers a custom script each time a StartUp, ShutDown, Freeze, Thaw, or DiskLow event occurs.	page 168
`iwatsub.ipl`	Triggers a custom script each time a file or a directory is submitted.	page 169
`iwattrib.exe`	Returns metadata information on any object in the TeamSite server, including all branches, workareas, editions, staging areas, files, directories, and symlinks.	page 79
`iwatunlock.ipl`	Triggers a custom script each time a file or a directory is unlocked.	page 170
`iwatupdate.ipl`	Triggers a custom script each time an update event occurs (Copy or Get Latest).	page 171
`iwcallback.exe`	Tells the TeamSite server that the program corresponding to an external or cgi task is finished. Passes a return code to the server.	page 89
`iwcat.exe`	Displays any version of a (text) file in TeamSite.	page 90
`iwchgrp.exe`	Changes the group of a TeamSite workarea or branch.	
`iwckrole.exe`	Checks whether or not a user can log in with a particular role.	page 91
`iwcmp.exe`	Compares any two TeamSite areas and returns a list of differences.	page 92
`iwconfig.exe`	Reads or writes to TeamSite's main configuration file.	page 28
`iwconvert`	Converts old-format (TeamSite 4.5.x and 5.0.x) backing stores to the new high-performance backing store format.	page 29
`iwconvertserver`	Supports the backing store conversion GUI by communicating with remote TeamSite servers and invoking `iwconvert` on behalf of the GUI.	page 32
`iwdecode.exe`	Decodes HTML-encoded %xx lines to ASCII lines.	page 94
`iwdeploy.exe`	Deploys website content to the production server.	page 95

`iwencode.exe`	Encodes ASCII lines to HTML-encoded %xx lines.	page 96
`iwevents.exe`	Returns a history of submissions or updates of a workarea.	page 97
`iwextattr.exe`	Sets extended attributes on a file.	page 100
`iwfreeze.exe`	Freezes and unfreezes all system writes.	page 33
`iwfsck.exe`	Diagnoses backing store problems.	page 35
`iwfsfix.exe`	Fixes problems found by `iwfsck`.	page 38
`iwfsshrink.exe`	Finds and removes duplicate data in the backing store.	page 40
`iwgetelog.exe`	Returns location of TeamSite events log.	page 42
`iwgetfilejobs.exe`	Returns a list of associated workflow job and task IDs for a file.	page 43
`iwgethome.exe`	Returns location of TeamSite program files.	page 44
`iwgetlocation.exe`	Returns the locations of various TeamSite log and configuration files.	page 45
`iwgetmount.exe`	Returns location of TeamSite mount point	page 47
`iwgetstore.exe`	Returns location of TeamSite backing store.	page 48
`iwgettrace.exe`	Returns location of TeamSite trace logs.	page 49
`iwgetwfobj.exe`	Prints the state of any part of the server workflow subsystem in XML.	page 102
`iwinvokejob.exe`	Starts a job whose instance has already been created on the TeamSite server.	page 109
`iwjobc.exe`	Creates a job instance (based on a job specification file) on the TeamSite server.	page 110
`iwjobvariable.exe`	Manipulates workflow variables.	page 111
`iwlasted.exe`	Returns the name of the last published edition on a branch.	page 112
`iwlist.exe`	Returns a list of all areas contained by a parent areas. For example, `iwlist` returns all editions, workareas, and branches contained on a parent branch.	page 113
`iwlistlocks.exe`	Lists the locks and assignments in a workarea or branch.	page 115

INTERWOVEN

`iwlistmod.exe`	Lists all modified files and directories in a specified area.	page 116
`iwlock.exe`	Locks any file or directory in a specified area.	page 117
`iwlockinfo.exe`	Provides detailed information on any lock.	page 118
`iwlsat.ipl`	Lists registered `iwat` programs.	page 172
`iwmigrate.exe`	Similar to `iwconvert` except that it accepts new-format backing stores (those created with `iwstoreadmin`, or converted with `iwconvert`) as its source.	page 51
`iwmkbr.exe`	Creates a new TeamSite branch.	page 120
`iwmkwa.exe`	Creates a new TeamSite workarea.	page 121
`iwnexted.exe`	Returns the name that would be autogenerated for the next edition on a branch.	page 122
`iwprefconv.exe`	Copies existing homepage information from the iwprofiles directory to the entity database.	page 53
`iwproxy.exe`	Invokes a proxy server.	page 54
`iwprv.exe`	Allows users to identify whether any file or directory in any workarea is marked private and automatically mark any file as either private or public.	page 123
`iwpublish.exe`	Publishes the staging area.	page 124
`iwqueryjobs.exe`	Prints a list of overall job information based on a query from stdin.	page 125
`iwquerytasks.exe`	Prints a list of job task information based on a query from stdin.	page 126
`iwrecentusers.exe`	Displays a list of everyone who has used TeamSite since the last time the TeamSite server was started, and a timestamp of each user's most recent TeamSite operation.	page 55
`iwrename.exe`	Renames a file, directory, workarea, edition, or branch.	page 128
`iwreset.exe`	Rereads TeamSite configuration files.	page 56
`iwretryjobop.exe`	Retries submission or task update after resolution of conflicts that originally prevented those operations.	page 129

`iwrevert.exe`	Revert to a previous version of a file.	page 130
`iwrlog.exe`	Shows a revision log for a TeamSite file.	page 131
`iwrmat.ipl`	Removes registered `iwat` programs.	page 173
`iwrmbr.exe`	Deletes a TeamSite branch and all contained areas (workareas, staging areas, and editions).	page 133
`iwrmed.exe`	Deletes an edition.	page 134
`iwrmjob.exe`	Unconditionally removes an entire job instance from the TeamSite server.	page 135
`iwrmtaskfile.exe`	Removes a file from a job task.	page 136
`iwrmtaskgroup.exe`	Removes groups from group tasks.	page 137
`iwrmtaskuser.exe`	Removes users from group tasks.	page 138
`iwsetjobdescription.exe`	Changes the description of a job.	page 140
`iwsetjobowner.exe`	Changes the owner of a job.	page 141
`iwsettaskattrib.exe`	Sets a new value for task attributes that have true or false values.	page 142
`iwrmuser.ipl`	Removes a user from TeamSite.	page 57
`iwrmwa.exe`	Deletes a TeamSite workarea.	page 139
`iwsi.ipl`	Collects system state data.	page 58
`iwstat.exe`	Returns current system activity.	page 60
`iwstoreadm`	Activates or deactivates a backing store.	page 62
`iwsubmit.exe`	Submits any TeamSite workarea, file, or directory to the staging area.	page 143
`iwtaketask.exe`	Assigns a shared task to a single user.	page 145
`iwtaskselect.exe`	Marks a user's job task finished and selects a successor task.	page 146
`iwteamsite.exe`	Installs or uninstalls the `iwtock` watchdog daemon.	page 64
`iwtestcfg.exe`	Returns the operation that will be performed upon a file at submission time.	page 65
`iwundochoice.exe`	Reverses a user-chosen transition from a user task or group task.	page 147
`iwunlock.exe`	Unlocks any file or directory.	page 148

INTERWOVEN

iwupdate.exe	Updates a TeamSite workarea with the version of a file or directory in the staging area.	page 149
iwuser	Enables you to manipulate TeamSite user information.	page 66
iwversion.exe	Returns current TeamSite release information.	page 69
iwvpath.exe	Prints all or parts of the version path of a specified object.	page 150

Appendix B

Sample Command Trigger Scripts

This appendix contains the following sample command trigger Perl scripts:

- `email_to.ipl` – sends email to a list of users when a triggering event occurs.
- `replicate_tobranch.ipl` – replicates submitted files and directories from a workarea to the staging area of a specified branch.

Email Notification Script

The following Perl script, `email_to.pl`, sends email to a specified list of users when an triggering event occurs. Usage and syntax are explained in comment lines.

```
#!/usr/local/bin/perl
# email_to.ipl
#
# Usage:
#       iwatasgn email_to.ipl tolist
#       iwatlock email_to.ipl tolist
#       iwatmkbr email_to.ipl tolist
#       iwatmkwa email_to.ipl tolist
#       iwatpub email_to.ipl tolist
#       iwatrmbr email_to.ipl tolist
#       iwatrmed email_to.ipl tolist
#       iwatrmwa email_to.ipl tolist
#       iwatsub email_to.ipl tolist
#       iwatunlock email_to.ipl tolist
#
# (Unix)
```

```
# Example : iwatpub email_to.pl 'mktg,staff,$IW_EVENTUSER'
# Example : iwatasgn email_to.pl '$IW_ASSIGNEE,$IW_EDITOR'
# Example : iwatsub email_to.pl '$IW_EVENTUSER,mktg'
#
# (NT)
# Example : perl iwatpub perl email_to.pl mktg,staff,$IW_EVENTUSER
# Example : perl iwatasgn perl email_to.pl $IW_ASSIGNEE,$IW_EDITOR
# Example : perl iwatsub perl email_to.pl $IW_EVENTUSER,mktg
#

$| = 1;

use Config;
if ($Config{osname} eq "MSWin32") {
        $MAILER = 'blat.exe - -t';
} else {
        $MAILER = '/usr/lib/sendmail';
}

# Change $DEFDOMAIN to be default domain suffix for user email
# $DEFDOMAIN = '@YOURDOMAIN.NET';

# $FROMUSER is added to From: line in emails
$FROMUSER = 'teamsite (TeamSite notifier)';

# $SIGNATURE is added to end of messages - signature string
$SIGNATURE = "\n\nTeamSite is brought to you by Interwoven.";

# Preprocess path to include iw-home in case script is called from
# environment without path initialization

$IW_HOME = `iwgethome`; chop $IW_HOME;
if ( $ENV{'PATH'} !~ m|$IW_HOME/bin| ) {
    $ENV{'PATH'} .= ":$IW_HOME/bin";
}

# Add $IW_HOME/lib/perl5 to Perl module include path
push @INC, "$IW_HOME/iw-perl/lib";

use File::Basename;
$MYNAME = basename($0, ".pl" );
```

```perl
use Env qw(PATH IWDEBUGERR IWDEBUGMAIL IW_EVENT IW_TIMESTAMP
        IW_EVENTUSER IW_EVENTROLE IW_AREA IW_FILEPATH IW_OBJID
        IW_ASSIGNEE IW_COMMENTS);

# $IWDEBUGERR true allows system calls to print errors to STDERR
# $IWDEBUGERR=1;

# $IWDEBUGMAIL true redirects output to STDOUT instead of mail
# program
# $IWDEBUGMAIL=1 unless defined($IWDEBUGMAIL);


# Redirect STDERR and set up signal
# handler to restore STDERR for final error message

$IWDEBUGERR=O unless defined($IWDEBUGERR);
if (! $IWDEBUGERR) {
    open(SAVESTDERR, ">&STDERR") || die "Can't save STDERR, $!\n";
    open(STDERR,">/dev/null") || die "Can't open STDERR to /dev/
        null\n";

    $SIG{__DIE__} = sub {
        open(STDERR, ">&SAVESTDERR");
        die "$MYNAME: " . $_[0];
    };
}
# Mail programs - uses reference to hash with keys being email
# users

sub MailTolist {
    my $maillist_p = $_[O];
    my $maillist = join ",", map "$_$DEFDOMAIN", keys %$maillist_p;
    return $maillist;
}
sub MailTo {
    my $maillist = MailTolist($_[O]);
    my $oldfh;

    return unless $maillist;

unless ($IWDEBUGMAIL) {
```

```perl
        open (MAIL, "| $MAILER $maillist")
            or die "Couldn't open pipe to $MAILER $maillist\n";
        $oldfh = select MAIL;
    }
    else {
        print "Pipe to $MAILER $maillist\n";
    }

    # Insert mail header
    print "From: $FROMUSER\n";
    print "To: $maillist\n";
    print "Reply-To: $FROMUSER\n";

    if ( $IW_EVENT eq "AssignFile" ) { &MailAssign; }
    elsif ( $IW_EVENT eq "ApproveFile" ) { &MailApprove; }
    elsif ( $IW_EVENT eq "RejectFile" ) { &MailReject ; }
    elsif ( $IW_EVENT eq "UnassignFile" ) { &MailUnassign; }
    elsif ( $IW_EVENT eq "MarkDoneFile" ) { &MailMarkDone; } ;

print "$IW_COMMENTS\n";

    &DumpAll;

    print "$SIGNATURE\n";

    unless ($IWDEBUGMAIL) {
        close (MAIL);
        select $oldfh;
    }
}

sub MailAssign {
    print "Subject: File assigned to $IW_ASSIGNEE: $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:             $IW_BRANCH\n";
    print "Workarea:           $IW_AREA\n";
    print "Workarea owner:     $IW_EDITOR\n";
    print "Assigned by user:   $IW_EVENTUSER\n";
    print "File path:          $IW_FILEPATH\n";
    print "\nAssignment comments:\n" . "-" x 60 . "\n";
}
```

```perl
sub MailApprove {
    print "Subject: File approved by $IW_EVENTUSER: $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:             $IW_BRANCH\n";
    print "Workarea:           $IW_AREA\n";
    print "Workarea owner:     $IW_EDITOR\n";
    print "Approved by:        $IW_EVENTUSER\n";
    print "File path:          $IW_FILEPATH\n";
    print "\nApproval comments:\n" . "-" x 60 . "\n";
}

sub MailReject {
    print "Subject: File rejected by $IW_EVENTUSER: $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:             $IW_BRANCH\n";
    print "Workarea:           $IW_AREA\n";
    print "Workarea owner:     $IW_EDITOR\n";
    print "Rejected by:        $IW_EVENTUSER\n";
    print "File path:          $IW_FILEPATH\n";
    print "\nRejection comments:\n" . "-" x 60 . "\n";
}

sub MailUnassign {
    print "Subject: File unassigned by $IW_EVENTUSER:
            $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:             $IW_BRANCH\n";
    print "Workarea:           $IW_AREA\n";
    print "Workarea owner:     $IW_EDITOR\n";
    print "Unassigned by:      $IW_EVENTUSER\n";
    print "File path:          $IW_FILEPATH\n";
    print "\nUnassignment comments:\n" . "-" x 60 . "\n";
}

sub MailMarkDone {
    print "Subject: File marked done by $IW_EVENTUSER:
            $IW_FILEPATH\n\n";
    print "Timestamp:          $IW_TIMESTAMP\n";
    print "Branch:             $IW_BRANCH\n";
    print "Workarea:           $IW_AREA\n";
    print "Workarea owner:     $IW_EDITOR\n";
```

```perl
    print "Marked done by:        $IW_EVENTUSER\n";
    print "File path:             $IW_FILEPATH\n";
    print "\nMark Done comments:\n" . "-" x 60 . "\n";
}
sub DumpAll {
        @vars = qw(
                  IW_TIMESTAMP IW_EVENTUSER IW_EVENTROLE IW_EVENT
                  IW_VPATH IW_EVENTLINE IW_BRANCH IW_EDITION
                  IW_STAGINGAREA IW_WORKAREA IW_AREA IW_FILE
                  IW_FILEPATH IW_OBJID IW_ASSIGNEE IW_EDITOR
                  IW_COMMENTS IW_SUBMITID);
        for ($i=0; $i<=$#vars; $i++) {
                $v = $vars[$i];
                if (! $ENV{$v}) { next; }
                print "$v : $ENV{$v}\n";
        }
}
# MAIN code
#
# Grab the mail targets...
#
foreach $mailDst (split(",", $ARGV[0])) {
        ##print $mailDst;
        if ($mailDst =~ /^\$/) {
                $mailDst =~ s/^\$//;
                $mailDst = $ENV{$mailDst};
                ##print "changed to $mailDst\n";
        }
        $emaillist{$mailDst} = 1;
}

MailTo(\%emaillist);

# -w mode can be a bit noisy for testing - eliminate unused var
# warnings
sub fakewarnings {
    my $i1 = \$DEFDOMAIN;
    my $i2 = \$SAVESTDERR;
    my $i3 = \$objid;
}
```

Replication Script

The following Perl script, `replicate_tobranch.pl`, replicates files and directories to the staging area of a specified branch. Replication is triggered whenever the files or directories are submitted to their original workarea. This script is designed for use with the `iwatsub` command trigger and is typically used to integrate content from one or more sub-branches back to a parent branch. Syntax and an example are included in comment lines. Due to this document's formatting constraints, some individual lines are depicted below as two or lines. These lines are indicated by a large bold-face exclamation point (**!**). A line ending with this character should be joined with the next line to form a single line of code.

```perl
#!/usr/local/bin/perl
#replicate_tobranch.pl
##This program replicates submitted files/dirs from a workarea
##to the staging area of a specified branch. This can be useful
##when integrating content from subbranches back to parent
##branches.
##Example:
##Given Teamsite directory layout:
##main->dev1
##        EDITION
##        STAGING
##        WORKAREA
##        ->dev2
##            EDITION
##            STAGING
##            WORKAREA
##              user1
##                file1
##
##> iwatsub replicate_tobranch.pl dev1
##
##When file1 is submitted to dev2 STAGING, a submit event
##will be generated, and file1 will also get submitted to
##STAGING in dev1.
push @INC, "/usr/local/lib/perl5";
```

INTERWOVEN

```perl
#get argument as the pathname for download directory

if ($#ARGV >=0) {
    $submitto_branch="//IWSERVER/default/main/". @ARGV[0];
} else {
    $submitto_branch="//IWSERVER/default/main/acmetest/A"; #default
}

#Assume the replicated workarea is called wa1
$replicate_workarea="$submitto_branch/WORKAREA/wa1";

chomp ($iwhome = `/usr/bin/iwgethome`);
$iwhomebin=$iwhome . "/bin";

# environment variables available
$event = $ENV{'IW_EVENT'};          # should always be "Submit"
$timestamp = $ENV{'IW_TIMESTAMP'};
$user = $ENV{'IW_EVENTUSER'};
$role = $ENV{'IW_EVENTROLE'};
$objid = $ENV{'IW_SUBMITID'};
$path = $ENV{'IW_WORKAREA'};

# testing only, get the last event, assume it's submit for now

if ($event eq "") {
  $eventlog=`$iwhomebin/iwgetelog`;
  chop($eventlog);
  $tailevent=`grep Submit $eventlog |tail -1`;

  @tailevent=split("\t",$tailevent);

  $event=@tailevent[3];
  $timestamp = @tailevent[0];
  $user = @tailevent[1];
  $role = @tailevent[2];
  $objid = @tailevent[5];
  chop($objid);
  $path = @tailevent[4];
}

#
```

```
#iwlist makes sure the branch is valid
#

$existbranch=`$iwhomebin/iwlist $submitto_branch 2>&1`;
chop($existbranch);

if ($existbranch =~ /ERROR/) {  #branch does not exist
    print STDERR ("$submitto_branch is not a valid branch.\n");
    exit(0);
}

#
#iwlist makes sure the replicated workarea exists in
#submitto_branch
#
$existreparea=`$iwhomebin/iwlist $replicate_workarea 2>&1`;
chop($existreparea);

if ($existreparea =~ /ERROR/) {  #reparea does not exist
    print STDERR ("$replicate_workarea need to exist.\n");
    exit(0);
}
```

INTERWOVEN

```
## get submit comment info and submit info field
print "GETTING SUBMIT COMMENTS\n";


$submit_cmt = `$iwhomebin/iwattrib -c submitevent -o $objid $path!
submit_cmt`;

$submit_info = `$iwhomebin/iwattrib -c submitevent -o $objid $path!
submit_info`;
$submit_cmt =~ s/\s+$//;
$submit_info =~ s/\s+$//;

## print info for debugging
print "
For Debugging...
User    : $user ($role)
Area    : $path
Time    : $timestamp
Submit Info: $submit_info
Submit Cmt : $submit_cmt
ObjId   : $objid

";

#
#find parentbranch from the path in the submit event
#

($parentbranch,$pathend) = ($path =~ /(.*)\/WORKAREA\/(.*)/ );

print "$parentbranch\n";
```

```perl
#get the list of files/dirs submitted from iwevents CLT

unless(open(SUBLOG, "$iwhomebin/iwevents -s -n $objid |")) {
    print STDERR "Can't open iwevents: $!\n";
    return;
}

#can only update to a workarea, so in order to submit to a
#specific branch, first
#copy it to a workarea under that branch, then submit

        #get individual submitted file/dir name and description

        $submit_objs="";
        while (<SUBLOG>) {
            @result = split('\t', $_);
            $action = $result[1];
            $obj_type = $result[2];
            ($filename) = &URLDecode($result[3]);
            ($description) = &URLDecode($result[5]);
            $description =~ s/(\s)+/ /g;          # loose newlines/tabs
            if ($description eq "<none>") {
                $description = "[no comment specified]";
             }
            print "DEBUG:filename : $filename\n";
            print "DEBUG:obj_type : $obj_type $action\n";
            print "DEBUG:Individual Comment: $description\n\n";

            ##construct full path name for the updated file and
             ##to-replicate workarea

            $updateobj = $parentbranch . "\/STAGING" .$filename;
            $replicateobj=$replicate_workarea;

             print "DEBUG:iwupdate -r $updateobj
                    $replicateobj...\n";

            # update obj to the to-replicate workarea

            @updatertn=`iwupdate -r $updateobj $replicateobj`;
            print ("$iwhomebin/iwupdate return: @updatertn\n");
```

INTERWOVEN

```
         # construct submit strings <obj comment> pairs to be
          #used in iwsubmit

         $submit_objs = $submit_objs. " " . $replicateobj .
                           $filename . " '". "$description" . "'";
         }

         print ("DEBUG: submit_objs = $submit_objs\n");
         #Submit the newly copied objs

         $submitcmd ="$iwhomebin/iwsubmit -r -c \'$submit_cmt\' !
         -i \'$submit_info\' $submit_objs";
         print "$submitcmd\n";

     $submitrtn=`$submitcmd`;
     print "DEBUG: submit rtn=$submitrtn\n";

sub URLDecode
{
#Decode a URL encoded string or array of strings
#+ is converted to space
#%NN is converted from hex
    foreach (@_) {
        s/%0D%0A/ /g;
        tr/+/ /;
        s/%(..)/pack("c", hex($1))/ge;
    }
    @_;
}
```

Appendix C

Error Codes

This table lists all the possible error codes that can be returned by a TeamSite command-line tool. Most of them are standard errors, but there are also many TeamSite-specific error codes. The list of TeamSite-specific error codes starts on page 201.

#	Error	Description
General OS and NFS errors		
-1	psx_EOF	
0	psx_ENFS_OK (NFS error)	
0	psx_SUCCESS	
0	psx_YES	
1	psx_FAILURE	Generic failure.
2	psx_ENFS_NOENT (NFS error)	No such file or directory.
2	psx_ENOENT	No such file or directory.
3	psx_ESRCH	No such process.
4	psx_EINTR	Interrupted system call.
5	psx_EIO	I/O error.
5	psx_ENFS_IO (NFS error)	A hard error occurred when the operation was in progress.
6	psx_ENFS_NXIO (NFS error)	No such device or address.
6	psx_ENXIO	No such device or address.
7	psx_E2BIG	Argument list too long.
8	psx_ENOEXEC	Executable file format error.

INTERWOVEN

#	Error	Description
9	psx_EBADF	Bad file number.
10	psx_ECHILD	No children.
11	psx_EAGAIN	Resource temporarily unavailable.
12	psx_ENOMEM	Not enough core memory.
13	psx_EACCES	Permission denied.
13	psx_ENFS_ACCES (NFS error)	Permission denied.
14	psx_EFAULT	Bad address.
15	psx_ENOTBLK	Block device required.
16	psx_EBUSY	Mount device busy.
17	psx_EEXIST	Item already exists.
17	psx_ENFS_EXIST (NFS error)	Item already exists.
18	psx_ENFS_XDEV (NFS error)	Attempt to do a cross-device hard link.
18	psx_EXDEV	Cross-device link.
19	psx_ENFS_NODEV (NFS error)	No such device.
19	psx_ENODEV	No such device.
20	psx_ENFS_NOTDIR (NFS error)	Directory expected, but wasn't one.
20	psx_ENOTDIR	Directory expected, but wasn't one.
21	psx_EISDIR	Is a directory.
21	psx_ENFS_ISDIR (NFS error)	Is a directory.
22	psx_EINVAL	Invalid argument.
22	psx_ENFS_INVAL (NFS error)	Invalid argument or unsupported argument for an operation.
23	psx_ENFILE	File table overflow.
24	psx_EMFILE	Too many open files.
25	psx_ENOTTY	Inappropriate I/O control for device.
26	psx_ETXTBSY	Text file busy.

TeamSite Command-Line Tools

#	Error	Description
27	psx_EFBIG	File too large.
27	psx_ENFS_FBIG (NFS error)	File too large.
28	psx_ENFS_NOSPC (NFS error)	Insufficient space to complete operation.
28	psx_ENOSPC	Insufficient space to complete operation.
29	psx_ESPIPE	Illegal seek.
30	psx_ENFS_ROFS (NFS error)	File system is read-only or archive is frozen.
30	psx_EROFS	File system is read-only or archive is frozen.
31	psx_EMLINK	Too many links.
31	psx_ENFS_MLINK (NFS error)	Too many hard links.
32	psx_EPIPE	Broken pipe.
33	psx_EDOM	Math argument out of domain of function.
34	psx_ERANGE	Math result not representable.
35	psx_ENOMSG	No message of desired type.
36	psx_EIDRM	Identifier removed.
37	psx_ECHRNG	Channel number out of range.
38	psx_EL2NSYNC	UNIX boot level 2 not synchronized.
39	psx_EL3HLT	UNIX boot level 3 halted.
40	psx_EL3RST	UNIX boot level 3 reset.
41	psx_ELNRNG	Link number out of range.
42	psx_EUNATCH	Protocol driver not attached.
43	psx_ENOCSI	No CSI structure available.
44	psx_EL2HLT	UNIX boot level 2 halted.
45	psx_EDEADLK	Resource deadlock condition.
46	psx_ENOLCK	No record locks available.
47	psx_ECANCELED	Operation canceled.

INTERWOVEN

#	Error	Description
48	psx_ENOTSUP	Operation not supported.
Filesystem Quotas		
49	psx_EDQUOT	Disk quota exceeded.
Convergent Error Returns		
50	psx_EBADE	Invalid exchange.
51	psx_EBADR	Invalid request descriptor.
52	psx_EXFULL	Exchange full.
53	psx_ENOANO	No anode.
54	psx_EBADRQC	Invalid request code.
55	psx_EBADSLT	Invalid slot.
56	psx_EDEADLOCK	File locking deadlock error.
57	psx_EBFONT	Bad font file format.
Stream Problems		
60	psx_ENOSTR	Device not a stream.
61	psx_ENODATA	No data (for no delay I/O).
62	psx_ETIME	Timer expired.
63	psx_ENOSR	Out of streams resources.
64	psx_ENONET	Machine is not on the network.
65	psx_ENOPKG	Package not installed.
66	psx_EREMOTE	The object is remote.
67	psx_ENOLINK	The link has been severed.
68	psx_EADV	Advertise error.
69	psx_ESRMNT	Srmount error.
70	psx_ECOMM	Communication error on send.
71	psx_EPROTO	Protocol error.
72	psx_EBADRPC	RPC struct is bad; error checks failed.

#	Error	Description
73	psx_EDOTDOT	RFS specific error.
74	psx_EMULTIHOP	Multihop attempted.
77	psx_EBADMSG	Trying to read unreadable message.
78	psx_ENAMETOOLONG	Path name is too long.
79	psx_EOVERFLOW	Value too large to be stored in data type.
80	psx_ENOTUNIQ	Given log. name not unique.
81	psx_EBADFD	File descriptor invalid for this operation.
82	psx_EREMCHG	Remote address changed.
Shared Library Problems		
83	psx_ELIBACC	Can't access a needed shared library.
84	psx_ELIBBAD	Accessing a corrupted shared library.
85	psx_ELIBSCN	.lib section in a.out corrupted.
86	psx_ELIBMAX	Attempting to link in too many libraries.
87	psx_ELIBEXEC	Attempting to exec a shared library.
88	psx_EILSEQ	Illegal byte sequence.
89	psx_ENOSYS	Unsupported file system operation.
90	psx_ELOOP	Symbolic link loop.
91	psx_ERESTART	Restartable system call.
92	psx_ESTRPIPE	Pipe or FIFO doesn't sleep in stream head.
93	psx_ENOTEMPTY	Directory not empty.
94	psx_EUSERS	Too many users (for UFS).

INTERWOVEN

#	Error	Description
BSD Networking Software		
argument errors		
95	psx_ENOTSOCK	Socket operation on non-socket.
96	psx_EDESTADDRREQ	Destination address required.
97	psx_EMSGSIZE	Message too long.
98	psx_EPROTOTYPE	Protocol wrong type for socket.
99	psx_ENOPROTOOPT	Protocol not available.
Linux-specific		
117	psx_EUCLEAN	Structure needs cleaning.
118	psx_ENOTNAM	Not a XENIX named type file.
119	psx_ENAVAIL	No XENIX semaphores available.
120	psx_EISNAM	Is a named type file.
121	psx_EREMOTEIO	Remote I/O error.
BSD Networking Software		
120	psx_EPROTONOSUPPORT	Protocol not supported.
122	psx_EOPNOTSUPP	Operation not supported on socket.
123	psx_EPFNOSUPPORT	Protocol family not supported.
124	psx_EAFNOSUPPORT	Address family not supported by protocol family.
125	psx_EADDRINUSE	Address already in use.
126	psx_EADDRNOTAVAIL	Can't assign requested address.
Operational Errors		
127	psx_ENETDOWN	Network is down.
128	psx_ENETUNREACH	Network is unreachable.
129	psx_ENETRESET	Network dropped connection because of reset.
130	psx_ECONNABORTED	Software caused connection abort.

#	Error	Description
131	psx_ECONNRESET	Connection reset by peer.
132	psx_ENOBUFS	No buffers available.
133	psx_EISCONN	Socket is already connected.
134	psx_ENOTCONN	Socket is not connected.
143	psx_ESHUTDOWN	Can't send after socket shutdown.
144	psx_ETOOMANYREFS	Too many references: can't splice.
145	psx_ETIMEDOUT	Connection timed out.
146	psx_ECONNREFUSED	Connection refused.
147	psx_EHOSTDOWN	Host is down.
148	psx_EHOSTUNREACH	No route to host.
149	psx_EALREADY	Operation already in progress/done.
150	psx_EINPROGRESS	Operation now in progress.
151	psx_ESTALE	Stale NFS file handle.
301	psx_EPERM	No permission; or not super-user.
301	psx_ENFS_PERM (NFS error)	Not owner.
311	psx_EWOULDBLOCK	Berkeley sockets error - operation would block till completion.
FreeBSD-specific		
398	psx_EJUSTRETURN	(400-2) don't modify registration, just return.
473	psx_ERPCMISMATCH	RPC version wrong.
474	psx_EPROGUNAVAIL	RPC program not available.
475	psx_EPROGMISMATCH	Program version wrong.
476	psx_EPROCUNAVAIL	Bad procedure for program.
479	psx_EFTYPE	Inappropriate file type or format.
480	psx_EAUTH	Authentication error.
481	psx_ENEEDAUTH	Need authenticator.

INTERWOVEN

#	Error	Description
Linux-specific		
512	psx_ERESTARTSYS	Linux system startup error.
513	psx_ERESTARTNOINTR	Linux system startup error.
514	psx_ERESTARTNOHAND	Restart if no handler.
515	psx_ENOIOCTLCMD	No I/O control command.
SunOS4-specific		
567	psx_EPROCLIM	Too many processes.
581	psx_ERREMOTE	Object is remote.
OSF/AXP-specific		
688	psx_ECLONEME	Tells open to clone the device.
689	psx_EDIRTY	Mounting a dirty (unchecked) file system.
690	psx_EDUPPKG	Duplicate package name on install.
691	psx_EVERSION	Version number mismatch.
693	psx_ENOSYM	Unresolved symbol name.
695	psx_EFAIL	Cannot start operation.
697	psx_EINPROG	Operation (now) in progress.
698	psx_EMTIMERS	Too many timers.
700	psx_EAIO	Internal AIO operation complete.
723	psx_ESOFT	Indicates a correctable error.
724	psx_EMEDIA	Returned by a disk driver to indicate a hard ECC error or similar disk media failure.
725	psx_ERELOCATED	"Success" code indicating that a defect relocation request was performed successfully.
Misc. generic error numbers		
901	psx_EEXITING	Exiting process/thread.

TeamSite Command-Line Tools

#	Error	Description
905	psx_EIMPLEMENT	Function not yet implemented.
910	psx_EBUFTOOSMALL	Buffer is too small for output.
911	psx_ESTRTOOLONG	String argument is too long.
920	psx_ENOTFOUND	Object being looked up was not found.
925	psx_ENOUSER	No such user.
930	psx_ERPCFAILURE	RPC connection failure.
980	psx_ENOAUTHENDONE	No authentication was done.
981	psx_ENOAUTHORDONE	No authorization was done.
999	psx_NO	"No" is the response.
Interwoven-specific error codes		
2000	psx_EITERENDED	Iteration ended.
2001	psx_EITERNOENT	No iteration entry.
2003	psx_ENOBASEED	Invalid base edition for operation.
2004	psx_ENOSE	No server found.
2005	psx_ENOAR	No archive found.
2006	psx_ENOBR	No branch found.
2007	psx_ENOWA	No workarea found.
2008	psx_ENOED	No edition found.
2009	psx_ENOFSE	No file or directory found.
2010	psx_ENOPREVFSE	No previous file or directory found.
2011	psx_ENOWADIR	Workarea directory not found.
2012	psx_ENOEDDIR	Edition directory not found.
2013	psx_ENOSA	No staging area found.
2014	psx_ENOOP	No operation found.
2015	psx_ENOTAG	No tag found.
2016	psx_ECONFLICTS	Conflicts prevent operation.

INTERWOVEN

#	Error	Description
2017	psx_ECHECKEDOUT	Checked out files prevent submitting.
2018	psx_EWABRBAD	Workarea does not match branch.
2019	psx_EONLYED	Cannot destroy last remaining edition on branch.
2100	psx_EBADNAME	Illegal name given.
2102	psx_EBADVPATH	Bad version path.
2150	psx_EPRIVATE	Illegal operation on private file or directory.
2200	psx_ENOROLE	No such role.
2300	psx_EILOP	Illegal operator.
2301	psx_ENOATTRIB	No such attribute.
2400	psx_ERESERVED	Object is being reserved by someone else.
2401	psx_ESAMEFSE	File or directory is the same as the other version.
2402	psx_ENEWERFSE	File or directory is newer than the other version.
2403	psx_ENOUPDATE	No update performed.
2404	psx_ENOSUBMIT	No submit performed.
2405	psx_ENOCMPRS	Compressing workarea, staging area, or latest edition is not allowed.
2406	psx_ECMPRED	Edition is already compressed.
2407	psx_EUNCMPRED	Edition is already uncompressed.
2408	psx_EMULTIBR	More than one branch found.
2409	psx_EMULTIWA	More than one workarea found.
2410	psx_ENOGROUP	No such group.
2411	psx_EDELETED	File had been deleted.
2412	psx_EASSIGNED	Assigned files prevent submitting.
2413	psx_ENOTAUTHOR	Must be Author.

#	Error	Description
2414	`psx_EOLDERFSE`	File is older than the other version.
2415	`psx_ENOTWAMBR`	Not owner or member of workarea.
2416	`psx_EABORTED`	Operation aborted.
2417	`psx_ENEWERDELETION`	File or directory is a deletion of the other file or directory.
2418	`psx_ENOROLEAUTH`	Role is not authorized for this user.
2419	`psx_EWRONGUSERPWD`	Invalid username/password combination.
2420	`psx_ENODOMAIN`	Domain does not exist.
2421	`psx_ENOSADIR`	Staging area directory not found.
2422	`psx_ENOTSUSER`	No such TeamSite user.
ODBC errors		
21098	`psx_ECONN_INVALID_HANDLE`	ODBC code: invalid handle.
21099	`psx_ECONN_ERROR`	ODBC code: generic error.
21101	`psx_ECONN_SUCCESS_WITH_INFO`	ODBC code: success; more information is coming.
21199	`psx_ECONN_NEED_DATA`	ODBC code: need more data.
21200	`psx_ECONN_NO_DATA_FOUND`	ODBC code: no data found.
Interwoven Object Transfer Protocol		
21501	`psx_ECOP_ILLXPORT`	Illegal transport.
21502	`psx_ECOP_XPORTNOTAVAIL`	Transport not available.
21503	`psx_ECOP_NOSERVER`	Server not found.
21504	`psx_ECOP_BUFTOOSMALL`	Buffer too small.
21505	`psx_ECOP_NOTENOUGHDATA`	Not enough bytes.
21506	`psx_ECOP_BADBYTECOUNT`	Incorrect byte count.
21507	`psx_ECOP_ILLDATA`	Illegal data.
21508	`psx_ECOP_METHODNOTAVAIL`	Specified method not available.
21510	`psx_ECOP_VERNOTSUP`	COP version not supported.

INTERWOVEN

#	Error	Description
21512	`psx_ECOP_BADMSGINDEX`	Incorrect message index.
22001	`psx_ESC_FAILURE`	General Special Channel failure.
22004	`psx_ESC_BADKEY`	Bad Special Channel key.
22006	`psx_ESC_NOCHANNEL`	No Special Channel.
22008	`psx_ESC_NODIR`	No Special Channel directory.
22010	`psx_ESC_CANTCREATE`	Cannot create Special Channel.
NFS errors		
100063	`psx_ENFS_NAMETOOLONG`	File name too long.
100066	`psx_ENFS_NOTEMPTY`	Directory not empty.
100069	`psx_ENFS_DQUOT`	Disk quota exceeded.
100070	`psx_ENFS_STALE`	Invalid file handle.
100071	`psx_ENFS_REMOTE`	The file handle given in the arguments referred to a file on a non-local file system on the server.
100099	`psx_ENFS_WFLUSH`	The server's write cache used in the "WRITECACHE" call got flushed to disk.
110001	`psx_ENFS_BADHANDLE`	Illegal NFS file handle. The file handle failed internal consistency checks.
110002	`psx_ENFS_NOT_SYNC`	Update synchronization mismatch was detected during a SETATTR operation.
110003	`psx_ENFS_BAD_COOKIE`	READDIR or READDIRPLUS cookie is stale.
110004	`psx_ENFS_NOTSUPP`	Operation is not supported.
110005	`psx_ENFS_TOOSMALL`	Buffer or request is too small.
110006	`psx_ENFS_SERVERFAULT`	An error occurred on the server which does not map to any of the legal NFS version 3 protocol error values
110007	`psx_ENFS_BADTYPE`	An attempt was made to create an object of a type not supported by the server.

#	Error	Description
110008	psx_ENFS_JUKEBOX	The server initiated the request, but was not able to complete it in a timely fashion.

TeamSite Command-Line Tools

Index

INTERWOVEN